308 ---- Q199808130001
Scientific Notebook # 234

21
300

R

Dr. David Ferrill Todd Lewis
Joshua Buckner; SWB

The Boorum & Pease® Quality Guarantee

The materials and craftsmanship that went into this product are of the finest quality. The pages are thread sewn, meaning they're bound to stay bound. The inks are moisture resistant and will not smear. And the uniform quality of the paper assures consistent rulings, excellent writing surface and erasability. If, at any time during normal use, this product does not perform to your expectations, we will replace it free of charge. Simply write to us:

Boorum & Pease Company
71 Clinton Road, Garden City, NY 11530
Attn: Marketing Services

Any correspondence should include the code number printed at the bottom of this page as well as the book title stamped at the bottom of the spine.

One Good Book Deserves Many Others.

Look for the complete line of Boorum & Pease® Columnar, Journal, and Record books. Custom-designed books also available by special order. For more information about our Customized Book Program, contact your office products dealer. See back cover for other books in this series.

Made in U.S.A.

date: June 12, 1997 Joshua William Backner

Initial Entry

Title: 3d stress Code Modification

Persons Involved: Joshua Backner

Objectives: To add features to the 3dstress application as needed
See also the SRD and the following wish list.

Special Training: experience with C / C++ programming languages and familiarity with the Unix envivanment

Version Control: RCS

Platform: SGI IndigoZ Impact

Source: /usr/people/3dstress/3d/src on yosemite.cnwra.swri.edu

Language: C and C++

Compiler: Irix 5.3 cc

Initial Entry Continued

3DStress V2.0 Alpha Wishlist & Refinements

[redacted], 1997

June 12, 1997

General:

1. Enable heterogeneous stress field entry such as output from UDEC (3D) or tool for building 2D and 3D stress tensor coverages.
2. Enable user entry of stress conditions as a function of depth.
3. Add option to automatically tile (arrange) windows on screens so windows do not overlap.
4. Port to Sun Ultra platform.

Plot Tendency Window:

1. Compute angular mismatches
2. Enable stress inversion computations – computed stress conditions from slip vectors and fault orientations.
3. Compute variance (uncertainty analysis) of tendency computations.
4. Use pre–defined 3D volume stress state to compute slip tend.

Sliders Window:

1. User specified units for magnitudes (ie. MPa, lbs/gal...) and compute shear and normal stresses in specified units.
2. Add a units conversion tool for converting stress magnitudes.

Graph Window:

No modifications.

3Dview Window:

1. Enable user to slice display along X–Y–Z axes or to specify range values along the axes.

Map Window:

1. Option to read Landmark Zmap and MVE VBL line files.
2. Display map coordinates for mouse cursor in lower portion of window.
3. Display a map–registered bitmap background image.
4. Enable user entry of a 2D map specifying the heterogeneous stress conditions over the map region. User draws polygons and specifies stress conditions for polygon regions.
5. Enable a range of dip values (two sliders) for color lines by dip option.
6. Enable user to specify the dip of each 2D fault trace.

3Dsurface Window:

No modifications.

Joshua Willcox Haimson

Joshua Willcox Haimson

General:

HIGH	Make sure that new fileSystem is fully functional.
HIGH	*Get all of the quick key functions in the graphics windows (ie F12..etc) refer to the new fileSystem.*
MEDIUM	Get browse buttons to work again.
MEDIUM	Read and write different file formats. This also needs to be done for color coverages to vbl.
MEDIUM	Look into another way of saving the contents of each graphics screens so that the save dialog box won't be saved with the image.
MEDIUM	When data files contain data that has around 8 decimal places the data becomes rounded when displayed. The problem seems to exist with OpenGL. Look at pave1*.lin for instances.
LOW	Redo how VBL files are read in (not through old linFileClaass).
LOW	3DStress binary file.,
LOW	Lighting needs to be worked on.
LOW	*GUI should be redone to use pixmaps on buttons instead of words.*
LOW	Let anti–aliasing be a universal toggle.
LOW	Update all the quick help buttons on each window module to contain more information.
LOW	Make sure that all windows now use the symbol for sigma instead of s1, s2, & s3.
LOW	Have variable fonts in the opengl window like in the motif windows. I.E. let the user set how large the text can be (should be integrated with the motifFontObj).
LOW	*When changing the background color from black to white, not all of the graphics windows seem to change.*
LOW	When a graphics window is minimized you should be able to maximize when you press the corrisponding button the main controler.
LOW	Any of the old file classes need to be removed from the code completely.
LOW	When controler is minimized all windows should be unmanaged.

Plot Tendency Window:

MEDIUM	Use symbols in place of words where possible, such as the equation being computed.
LOW	Still have problems when resizing the window that things are not drawn properly.
LOW	In options menu allow for the adjust strike and dip buttons to be able to be held down and take effect instead of continually clicking.
LOW	Use full azimuth notaion ie 010 degs, not simply 10 degs (Chevron)
LOW	Allow users to create own equations to be computed instead of just Dilation and Slip (Chevron).
LOW	Does Budhi have a way to compute TsMax for a given stress state? Otherwise, TsMax is sometimes 102% for step size = 10 deg.
LOW	Set up animation playback mode, read from text file: S1 S2 S3 rotx roty rotz flystrike fltdip sleepsec.

Sliders Window:

MEDIUM	Ability to load and save magnitudes does not fully work.
MEDIUM	Saving should also incorporate the fluidPressure and tensileStr

Graph Window:

MEDIUM	Change sigmas to the actual sigma symbol. Must look intro stroke fonts.

Options Window:

LOW	Rearrange items to take up less room. Only show thresholding sliders and lighting sliders when that mode is activated.

Joshua William Buckner

MohrCircle Window:

HIGH	Make sure when apply button is pressed, tensile str and fluid pressure are changed on the sliders tool, also make sure that the values are correct.
MEDIUM	Changed on graphics display tensile str to computed tensile str, so not to get confused with the tensile str on the sliders tool.
LOW	Options window needs to be redone
LOW	Needs information for quick help.
LOW	Ability to issuse some of the commands through the keyboard.
LOW	Display fluid pressure in graphics window.

Map Window:

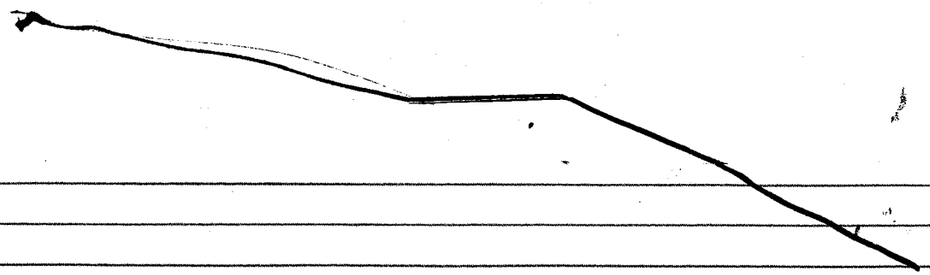
MEDIUM	Get fault builder to work correctly and be more user friendly. Also allow for saving to the fileSystem.
LOW	Toggle displaying of bounding box.

3Dsurface Window:

LOW	In options menu allow for the verticle exageration buttons to be able to be held down and take effect instead of continually clicking.

3Dview Window:

HIGH	*Allow for 3ds color coverage. This allows for saving of selected triangles while retaining their color to be reloaded back in as a coverage. Also to be exported to 3DMove.*
HIGH	When multiple files are loaded and only selected ones are removed, the bounding box will sometimes not resize to fit current files.
HIGH	*Picking does not allways work. When you load and remove several files there tends to be a problem selecting on some of the faults.*
HIGH	Bounding box does not allways resize to fit loaded files.
HIGH	Sometimes a redraw event isn't made when removing files.
MEDIUM	Get fault builder to work correctly and be more user friendly. Also allow for saving to the fileSystem.
MEDIUM	Able to export a scaled .cgm or other graphic file of the 3d image, project onto an arbitrary plane. Than can overlay the result with independently produced Allan maps.
LOW	*Export files as VRML.*
LOW	*Allow for 3D coverages with the 3Dviewer*
LOW	Experiment on how to display selected triangles while in solid mode.
LOW	Viewer options window is very large, allow for showing of portions.
LOW	Try using display lists for the bounding box.
LOW	Allow user to change the spacing between bounding box grid lines.
LOW	Normal and slip vectors should project from both sides (Chevron).
LOW	Show scaling (Chevron).
LOW	*Select all feature (Chevron).*
LOW	Toggle displaying of bounding box.

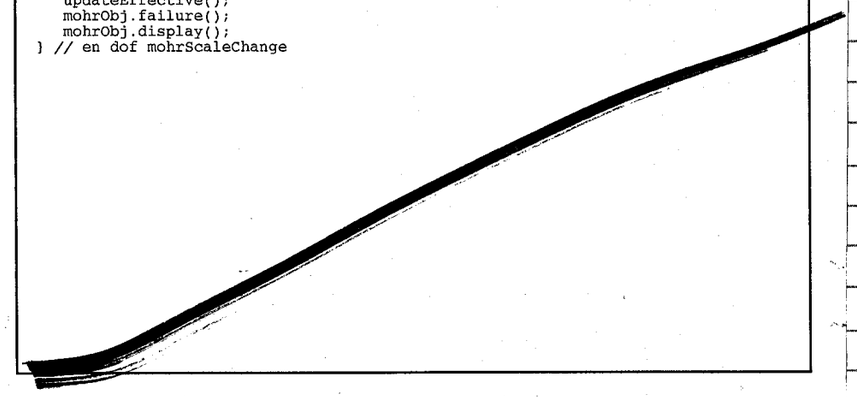


Initial Entry Ends; In process Entries begin

Joshua William Buckner

Jun 13 1997 13:50 **mohr.chg** Page 1

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//                Ask Robert Boenau what the hell these are.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigma
//                  that was changed in the Mohr graph options window
//                  by the user
//
//  Purpose:  This program updates the sigmas according to the settings
//            of the scales in the Mohr graph options window devoted to
//            the sigmas.
//
// Last Modified:  13 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below, mattr, holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   //  The variable below, whichSig, is the index of the sigma that
   //  was changed in the Mohr's options window.  Also, newValue holds
   //  the value that the user set to the sigma indicated by whichSig
   int
      whichSig = (int) clientData,
      newValue = (int) cbs->value;

   mohrObjAttrib->sigmas[whichSig] = newValue;

   //  Now we determine which sigma was changed and apply the
   //  relationship that was suggested to Dr. Ferrill.
   //  It was assumed that sigmas[1] is the verticle effective stress
   //  and that sigmas[3] is the horizontal effective stress?!
   //  Also, Poisson's ratio was taken to be 0.25
   if (whichSig == 1) // verticle stress was changed
      mohrObjAttrib->sigmas[3] = (0.25 / (1 - 0.25)) * newValue;
   else if(whichSig == 3) // horizontal stress was changed
      mohrObjAttrib->sigmas[1] = (1 - 0.25 / 0.25) * newValue;

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // en dof mohrScaleChange
```



```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              Ask Robert Boenau what the hell these are.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigma
//                  that was changed in the Mohr graph options window
//                  by the user
//
//  Purpose:  This program updates the sigmas according to the settings
//            of the scales in the Mohr graph options window devoted to
//            the sigmas.
//
// Last Modified:  16 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//---------------------------------------------------------------------------
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below, mattr, holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   //  The variable below, whichSig, is the index of the sigma that
   //  was changed in the Mohr's options window.  Also, newValue holds
   //  the value that the user set to the sigma indicated by whichSig,
   //  and delta holds the change in the sigma value.
   int
      whichSig = (int) clientData,
      newValue = (int) cbs->value,
      delta = mohrObjAttrib->sigmas[whichSig] - newValue;

   mohrObjAttrib->sigmas[whichSig] = newValue;

   //  Now we determine which sigma was changed and apply the
   //  relationship that was suggested to Dr. Ferrill.
   //  It was assumed that sigmas[1] is the verticle effective stress
   //  and that sigmas[3] is the horizontal effective stress?!
   //  Also, Poisson's ratio was taken to be 0.25
   if (whichSig == 1) // verticle stress was changed
      mohrObjAttrib->sigmas[3] += (0.25 / (1 - 0.25)) * delta;
   else if(whichSig == 3) // horizontal stress was changed
      mohrObjAttrib->sigmas[1] += (1 - 0.25 / 0.25) * delta;

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // end of mohrScaleChange
```



```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              Ask Robert Boenau what the hell these are.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigma
//                  that was changed in the Mohr graph options window
//                  by the user.  Scales in the Options window are
//                  changed.
//
//  Purpose:  This program updates the sigmas according to the settings
//            of the scales in the Mohr graph options window devoted to
//            the sigmas.
//
// Last Modified:  16 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//---------------------------------------------------------------------------
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   //  The variable below holds the member variables of the
   //  class responsible for the Mohr Graph's Options window
   MohrOptionType *optionsAttrib = mohrOptionObj.getAttributes();

   //  The variable below, whichSig, is the index of the sigma that
   //  was changed in the Mohr's options window.  Also, newValue holds
   //  the value that the user set to the sigma indicated by whichSig,
   //  and delta holds the change in the sigma value.
   int
      whichSig = (int) clientData,
      newValue = (int) cbs->value,
      delta = newValue - mohrObjAttrib->sigmas[whichSig],
      max = 0, min = 0,  // indecies of largest & smallest sigma values
      i,  // loop index
      oldOtherValue;  // holds value of sigma changed by formula

   // Search for the largest and smallest sigmas
   for(i = 1; i < 3; i++)
   {
      if (mohrObjAttrib->sigmas[i] >= mohrObjAttrib->sigmas[max])
         max = i;
      if (mohrObjAttrib->sigmas[i] <= mohrObjAttrib->sigmas[min])
         min = i;
   }

   if ( whichSig == max )
   {  // if the verticle stress was changed

      // change the horizontal stress using the little formula
      // saving the old value just in case
      oldOtherValue = mohrObjAttrib->sigmas[min];
      mohrObjAttrib->sigmas[min] += (0.25 / (1.0 - 0.25)) * delta;

      if(mohrObjAttrib->sigmas[min] > 100)
      { // if the change will exceed the bounds, then stop at bound
         mohrObjAttrib->sigmas[min] = 100;
         mohrObjAttrib->sigmas[max] += (100 - oldOtherValue) * ((1.0 - 0.25) / 0.2
5);

         // display the change in sigma on the scale bar in the options window
         XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[min]);
      }  // end if
```

Printed by jbuckner from yosemite


```
      else if(mohrObjAttrib->sigmas[min] < -100)
      { // if the change will exceed the bounds, then stop at bound
         mohrObjAttrib->sigmas[min] = -100;
         mohrObjAttrib->sigmas[max] += (-100 - oldOtherValue) * ((1.0 - 0.25) / 0.
25);

         // display the change in sigma on the scale bar in the options window
         XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[min]);
      }  // end else if
      else  // bounds are all right
      {
         // update the sigma changed by the user
         mohrObjAttrib->sigmas[max] = newValue;
      }  // end else

      // display the change in sigma on the scale bar in the options window
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min]);

   }  // end if

   else if ( whichSig == min )
   {  // if the horizontal stress was changed

      // change the verticle stress using the little formula
      // saving the old value just in case
      oldOtherValue = mohrObjAttrib->sigmas[max];
      mohrObjAttrib->sigmas[max] += ((1.0 - 0.25)/0.25) * delta;

      if(mohrObjAttrib->sigmas[max] > 100)
      { // if the change will exceed the bounds, then stop at bound
         mohrObjAttrib->sigmas[max] = 100;
         mohrObjAttrib->sigmas[min] += (100 - oldOtherValue) * ((1.0 - 0.25) / 0.2
5);

         // display the change in sigma on the scale bar in the options window
         XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min]);
      }  // end if
      else if(mohrObjAttrib->sigmas[max] < -100)
      { // if the change will exceed the bounds, then stop at bound
         mohrObjAttrib->sigmas[max] = -100;
         mohrObjAttrib->sigmas[min] += (-100 - oldOtherValue) * ((1.0 - 0.25) / 0.
25);

         // display the change in sigma on the scale bar in the options window
         XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min]);
      }  // end else if
      else  // bounds are all right
      {
         // update the sigma changed by the user
         mohrObjAttrib->sigmas[min] = newValue;
      }  // end else

      // display the change in sigma on the scale bar in the options window
      XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max]);

   }  // end else if

   else  // middle value changed
   {
      mohrObjAttrib->sigmas[whichSig] = newValue;
   }  // end else

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // end of mohrScaleChange
```


mohrOptionClass.hh: (lines 40 to 42) addition to struct MohrOptionType

```
   Widget   stressDependanceRadio,        //  radio buttons for specifying
            stressDependantRadio,         //  wether or not the stresses
            stressIndependantRadio;       //  are dependant on each other
```

mohrOptionClass.c++: added radio box but ran into problems

```
// define radio buttons for setting dependantcy of stresses on each other

   frame = XtVaCreateManagedWidget("frameStress", xmFrameWidgetClass,
                                   col2,
                                   XmNbackground, WIDGET_COLOR,
                                   NULL);

   Widget colStress = XtVaCreateWidget("colStressWidget",
                        xmRowColumnWidgetClass, frame,
                        XmNorientation, XmVERTICAL,
                        XmNnumColumns, 7,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

   Widget stressDependanceLabel = XtVaCreateManagedWidget("stressLable",
                         xmLabelWidgetClass, colStress,
                         XtVaTypedArg,
                         XmNlabelString, XmRString,
                         "Stress Mode", 12,
                         XmNalignment, XmALIGNMENT_CENTER,
                         XmNbackground, WIDGET_COLOR,
                         NULL);

   Widget seperatorStress = XtVaCreateManagedWidget("SeperatorStress",
                                    xmSeparatorGadgetClass,
                                    colStress, NULL);

   attributes.stressDependanceRadio = XmCreateRadioBox(colStress, "stressDependanc
eRadio", NULL, 0);
   XmChangeColor(attributes.stressDependanceRadio, WIDGET_COLOR);

   attributes.stressDependantRadio = XtVaCreateManagedWidget("stressDependantRadio
",
                                xmToggleButtonGadgetClass,
                                attributes.stressDependanceRadio,
                                XtVaTypedArg,
                                XmNlabelString, XmRString,
                                "Dependant Stresses", 19,
                                XmNselectColor, BUTTON_COLOR,
                                NULL);

   attributes.stressIndependantRadio = XtVaCreateManagedWidget("stressIndependantR
adio",
                                xmToggleButtonGadgetClass,
                                attributes.stressDependanceRadio,
                                XtVaTypedArg,
                                XmNlabelString, XmRString,
                                "Independant Stresses", 21,
                                XmNselectColor, BUTTON_COLOR,
                                NULL);

   if (mattr->stressMode == 1)
      XmToggleButtonSetState(attributes.stressIndependantRadio, TRUE, TRUE);
   else
      XmToggleButtonSetState(attributes.stressDependanceRadio, TRUE, TRUE);

// end of section for stress dependancy radio buttons
```

mohrOptionClass.c++: added call backs radio box

```
// Call backs to change the stress mode between dependant and independant
   XtAddCallback (attributes.stressDependantRadio, XmNvalueChangedCallback,
```

mohrOptionClass.c++: (lines 417 to 471)

```
// define radio buttons for setting dependantcy of stresses on each other

   frame = XtVaCreateManagedWidget("frameStress", xmFrameWidgetClass,
                                   col2,
                                   XmNbackground, WIDGET_COLOR,
                                   NULL);

   Widget colStress = XtVaCreateWidget("colStressWidget",
                           xmRowColumnWidgetClass, frame,
                           XmNorientation, XmVERTICAL,
                           XmNnumColumns, 7,
                           XmNpacking, XmPACK_TIGHT,
                           XmNbackground, WIDGET_COLOR,
                           NULL);

   Widget stressDependanceLabel = XtVaCreateManagedWidget("stressLable",
                            xmLabelWidgetClass, colStress,
                            XtVaTypedArg,
                            XmNlabelString, XmRString,
                            "Stress Mode", 12,
                            XmNalignment, XmALIGNMENT_CENTER,
                            XmNbackground, WIDGET_COLOR,
                            NULL);

   Widget seperatorStress = XtVaCreateManagedWidget("SeperatorStress",
                                       xmSeparatorGadgetClass,
                                       colStress, NULL);

   attributes.stressDependanceRadio = XmCreateRadioBox(colStress, "stressDependanc
eRadio", NULL, 0);
   XmChangeColor(attributes.stressDependanceRadio, WIDGET_COLOR);

   attributes.stressDependantRadio = XtVaCreateManagedWidget("stressDependantRadio
",
                                xmToggleButtonGadgetClass,
                                attributes.stressDependanceRadio,
                                XtVaTypedArg,
                                XmNlabelString, XmRString,
                                "Dependant Stresses", 19,
                                XmNselectColor, BUTTON_COLOR,
                                NULL);

   attributes.stressIndependantRadio = XtVaCreateManagedWidget("stressIndependantR
adio",
                                xmToggleButtonGadgetClass,
                                attributes.stressDependanceRadio,
                                XtVaTypedArg,
                                XmNlabelString, XmRString,
                                "Indepandant Stresses", 21,
                                XmNselectColor, BUTTON_COLOR,
                                NULL);

   if (mattr->stressMode == 1)
      XmToggleButtonSetState(attributes.stressIndependantRadio, TRUE, TRUE);
   else
      XmToggleButtonSetState(attributes.stressDependanceRadio, TRUE, TRUE);

// end of section for stress dependancy radio buttons
```

mohrOptionClass.c++: (lines 907 to 910)

```
 // Stress  dependency radio buttons
   XtManageChild(colStress);
   XtManageChild(attributes.stressDependanceRadio);
 // end Stress  dependency radio buttons
```


mohrClass.c++: (line 101) added initialization of a member variable
this ended the core dump of 18 June 97

```
   attributes.stressMode = 1;
```

added a variable (Poisson's and max, min scale) to mohrClass.hh

```
// Stress Dependency relate variables
     int          stressMode;    // boolean for stress Dependency
     int          stressDepRatio; // the degree to which stresses are related
     int          stressScaleMax;   // maximum value allowed on stress ratio sc
ale
     int          stressScaleMin;   // minimum value allowed on stress ratio sc
ale
// End of Stress Dependency related variables
```

added initialization of above to mohrClass.c++

```
// Stress related member variable initialization
   attributes.stressMode = 1;     // stresses are initially independent
   attributes.stressDepRatio = 25;  // Poisson's ratio is initially 25%
   attributes.stressScaleMax = 100; // Max Poisson's Ratio is 100%
   attributes.stressScaleMin = 1;   // Min Poisson's Ratio is 1%
// End of Stress related member variable initialization
```

added Widget declaration to mohrOptionClass.hh for Poisson Scale

```
   Widget   stressScale;                  //  Scale for specifiying Poisson's
```

added Widget code to mohrOptionClass.c++ for Poisson Scale

```
   // define a scale for the ratio between dependent stresses
   Widget stressScaleLabel = XtVaCreateManagedWidget("stressScaleLable",
                            xmLabelWidgetClass, col3,
                            XtVaTypedArg, XmNlabelString, XmRString,
                            "Dependent Stress Ratio", 23,
                            XmNalignment, XmALIGNMENT_CENTER,
                            XmNbackground, WIDGET_COLOR,
                            NULL);

   Widget sepStressScale = XtVaCreateManagedWidget("SepStressScale",
                                        xmSeparatorGadgetClass,
                                        col3, NULL);

   attributes.stressScale = XtVaCreateManagedWidget("W",
                            xmScaleWidgetClass, col3,
                            XmNorientation, XmHORIZONTAL,
                            XmNmaximum, mattr->stressScaleMax,
                            XmNminimum, mattr->stressScaleMin,
                            XmNvalue, 25,
                            XmNscaleMultiple, 1,
                            XmNshowValue, True,
                            XmNbackground, WIDGET_COLOR,
                            NULL);
   // end of define a scale for the ratio between Dependent stresses
```

added XtAddCallback to mohrOptionClass.c++ for Poisson Scale

```
   // Call back for the stress ratio scale
   XtAddCallback (attributes.stressScale, XmNdragCallback,
                  mohrStressScaleChange, (XtPointer)0);
```

added call back prototype for Poisson scale to mohrOptionCB.hh

```
   void mohrStressScaleChange(Widget, XtPointer, XtPointer);
```

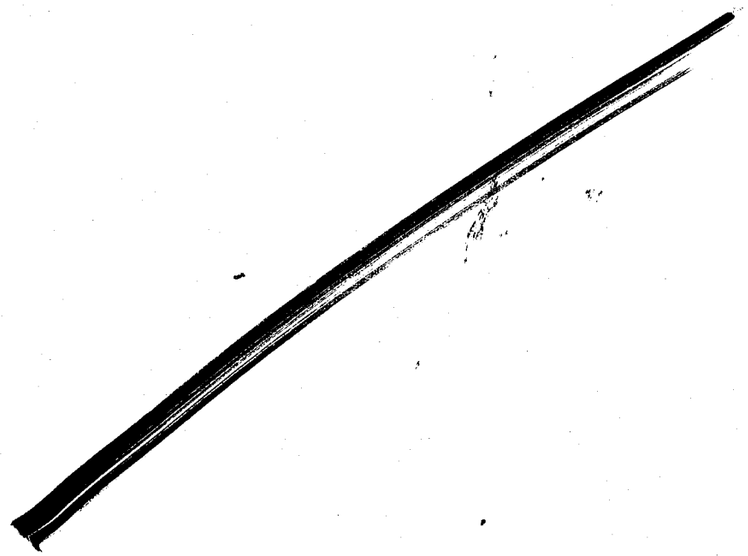
added call back definition for Poisson scale to mohrOptionCB.c++

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrStressScaleChange (not a member function)
//
```

Printed by jbuckner from yosemite


```
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//
//  State Changes:  mohrObj.attributes->stressDepRatio, updates the degree
//                  to which the stress are Dependent when in Dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of Dependency for Dependent stresses
//
// Last Modified:  19 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrStressScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   // update the ratio
   mohrObjAttrib->stressDepRatio =  (int) cbs->value;

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrStressScaleChange
```

mohrOptionCB.c++: Added member variable of mohrObj to calculation of stresses in the mohrScaleChange function. It is divided by 100 and then stored in the mohrScaleChange local variable called ratio.



mohrOptionCB.c++

changed all
```
   XmScaleSetValue(optionsAttrib->scale[?], mohrObjAttrib->sigmas[?]);
```
to
```
   XmScaleSetValue(optionsAttrib->scale[?], mohrObjAttrib->sigmas[?] * 1000);
```

Fixed the bugs in the mistaken index values of the XmScaleSetValue calls and added even more bounds checking inside the bounds checking:

```
   if ( whichSig == max )
   {  // if the verticle stress was changed

      // change the horizontal stress using the little formula
      // saving the old value just in case
      oldOtherValue = mohrObjAttrib->sigmas[min];
      mohrObjAttrib->sigmas[min] += (ratio / (1.0 - ratio)) * delta;

      if(mohrObjAttrib->sigmas[min] > 100)
      { // if the change will exceed the bounds, then stop at bound
         mohrObjAttrib->sigmas[min] = 100;
         mohrObjAttrib->sigmas[max] += (100 - oldOtherValue) * ((1.0 - ratio) /
ratio);

         // check bounds on other sigma
         if (mohrObjectAttrib->sigmas[max] > 100)
            mohrObjectAttrib->sigmas[max] = 100;
         if (mohrObjectAttrib->sigmas[max] < -100)
            mohrObjectAttrib->sigmas[max] = -100;

         // display the change in sigma on the scale bar in the options window
         XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max]
* 1000);
      }  // end if(mohrObjAttrib->sigmas[min] > 100)
      else if(mohrObjAttrib->sigmas[min] < -100)
      { // if the change will exceed the bounds, then stop at bound
         mohrObjAttrib->sigmas[min] = -100;
         mohrObjAttrib->sigmas[max] += (-100 - oldOtherValue) * ((1.0 - ratio)
/ ratio);

         // check bounds on other sigma
         if (mohrObjectAttrib->sigmas[max] > 100)
            mohrObjectAttrib->sigmas[max] = 100;
         if (mohrObjectAttrib->sigmas[max] < -100)
            mohrObjectAttrib->sigmas[max] = -100;
         // display the change in sigma on the scale bar in the options window

         XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max]
* 1000);
      }  // end else if(mohrObjAttrib->sigmas[min] < -100)
      else  // bounds are all right
      {
         // update the sigma changed by the user
         mohrObjAttrib->sigmas[max] = newValue;
      }  // end else

      // display the change in sigma on the scale bar in the options window
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min] * 1
000);
   }  // end if( whichSig == max )

   else if ( whichSig == min )
   {  // if the horizontal stress was changed

      // change the verticle stress using the little formula
      // saving the old value just in case
      oldOtherValue = mohrObjAttrib->sigmas[max];
      mohrObjAttrib->sigmas[max] += ((1.0 - ratio)/ratio) * delta;

      if(mohrObjAttrib->sigmas[max] > 100)
      { // if the change will exceed the bounds, then stop at bound
```

Printed by jbuckner from yosemite


```
            mohrObjAttrib->sigmas[max] = 100;
            mohrObjAttrib->sigmas[min] += (100 - oldOtherValue) * ((1.0 - ratio) /
 ratio);

            // check bounds on other sigma
            if (mohrObjectAttrib->sigmas[min] > 100)
               mohrObjectAttrib->sigmas[min] = 100;
            if (mohrObjectAttrib->sigmas[min] < -100)
               mohrObjectAttrib->sigmas[min] = -100;

            // display the change in sigma on the scale bar in the options window
            XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min]
* 1000);
         }  // end if(mohrObjAttrib->sigmas[max] > 100)

         else if(mohrObjAttrib->sigmas[max] < -100)
         { // if the change will exceed the bounds, then stop at bound
            mohrObjAttrib->sigmas[max] = -100;
            mohrObjAttrib->sigmas[min] += (-100 - oldOtherValue) * ((1.0 - ratio)
/ ratio);

            // check bounds on other sigma
            if (mohrObjectAttrib->sigmas[min] > 100)
               mohrObjectAttrib->sigmas[min] = 100;
            if (mohrObjectAttrib->sigmas[min] < -100)
               mohrObjectAttrib->sigmas[min] = -100;

            // display the change in sigma on the scale bar in the options window
            XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min]
* 1000);
         }  // end else if(mohrObjAttrib->sigmas[max] < -100)

         else  // bounds are all right
         {
            // update the sigma changed by the user
            mohrObjAttrib->sigmas[min] = newValue;
         }  // end else

         // display the change in sigma on the scale bar in the options window
         XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max] * 1
000);
      }  // end else if
```

mohrOptionCB.c++:

Greying out of Poisson's % scale when stresses are independent was added:

```
void mohrToggleStressMode(Widget, XtPointer clientData, XtPointer)
{
   // get the attributes of the current Mohr graph for changing
   MohrAttributeType *mohrAttrib = mohrObj.getAttributes();

   // get the attributes of the option window (widgets and such)
   MohrOptionType *optionAttrib = mohrOptionObj.getAttributes();

   // set the stress mode to the appropriate value (0 or 1)
   // stressMode will be 0 for dependent stresses and 1 for independent
   mohrAttrib->stressMode = (int)clientData;

   if (mohrAttrib->stressMode == 1)  // if stress is independent, grey out the
ratio scale
      XtSetSensitive(optionAttrib->stressScale, False);
   else  // if stress is dependent, activate the ratio scale
      XtSetSensitive(optionAttrib->stressScale, True);
}
```

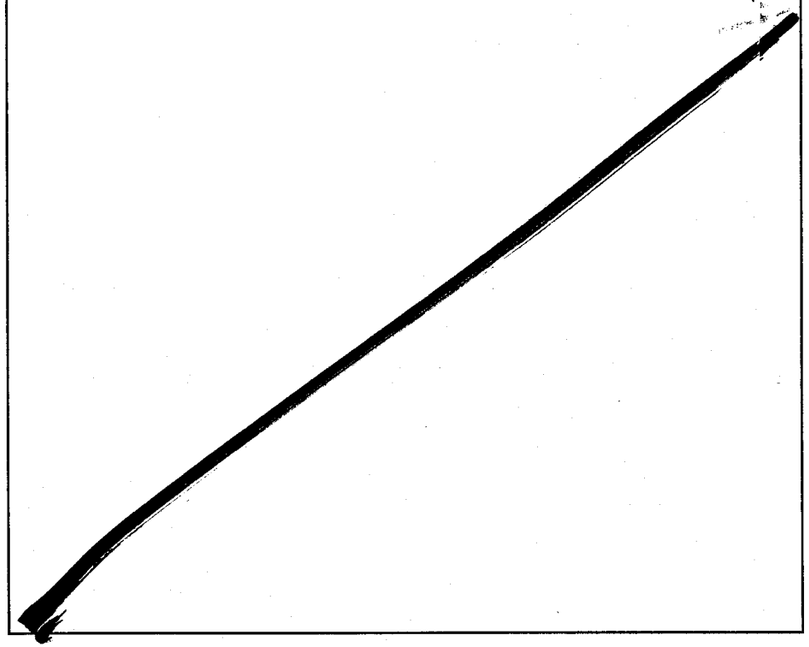
mohrOptionClass.c++

added greying of Poisson's % scale to initialization of the Options window:

```
// define a scale for the ratio between dependent stresses
```



```
Widget stressScaleLabel = XtVaCreateManagedWidget("stressScaleLable",
                        xmLabelWidgetClass, col3,
                        XtVaTypedArg, XmNlabelString, XmRString,
                        "Poisson's %", 12,
                        XmNalignment, XmALIGNMENT_CENTER,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

Widget sepStressScale = XtVaCreateManagedWidget("SepStressScale",
                              xmSeparatorGadgetClass,
                              col3, NULL);

attributes.stressScale = XtVaCreateManagedWidget("W",
                           xmScaleWidgetClass, col3,
                           XmNorientation, XmHORIZONTAL,
                           XmNmaximum, mattr->stressScaleMax,
                           XmNminimum, mattr->stressScaleMin,
                           XmNvalue, 25,
                           XmNscaleMultiple, 1,
                           XmNshowValue, True,
                           XmNbackground, WIDGET_COLOR,
                           NULL);

// Poisson's % scale should initially be inactive
XtSetSensitive(attributes.stressScale, False);
// end of define a scale for the ratio between dependent stresses
```



mohrRockInfoClass.hh:

```
   Created this file to produce a dialog box citing the source of the
   rock data appearing in the Mohr graph's option window.

#ifndef __MOHRROCKINFOCLASS_HH_
#define __MOHRROCKINFOCLASS_HH_

class MohrRockInfoClass
{
   public:
      MohrRockInfoClass(void);
      MohrRockInfoClass(Widget hook);
      ~MohrRockInfoClass(void);
      initialize(Widget hook);
   private:

      // Call backs
      void functionCloseButton(Widget, XtPointer, XtPointer);

      // Member Widgets
      Widget
         dialogForm, textArea, seperator, closeButton;
};

#endif
```

mohrRockInfoClass.c++:

```
   Created this file to produce a dialog box citing the source of the
   rock data appearing in the Mohr graph's option window.

// Standard C includes
#include <stdio.h>

// Motif includes
#include <Xm/RowColumn.h>
#include <Xm/PushBG.h>
#include <Xm/PushB.h>
#include <Xm/Label.h>
#include <Xm/MessageB.h>
#include <Xm/Form.h>
#include <Xm/SelectioB.h>
#include <Xm/Frame.h>
#include <Xm/TextF.h>
#include <Xm/Text.h>

//Application includes
#include "mohrRockInfoClass.hh"
#include "mohrOptionObj.hh"
#include "mohrOptionClass.hh"
#include "widgetGlobals.hh"
#include "motifFontObj.hh"

MohrRockInfoClass::MohrRockInfoClass(void)
{
   // This space intentionally left blank
}

MohrRockInfoClass::MohrRockInfoClass(Widget hook)
{
   MohrRockInfoClass::initialize(hook);
}

MohrRockInfoClass::~MohrRockInfoClass(void)
{
   // This space intentionally left blank
}

// Call Backs
```



```
void functionCloseButton(Widget, XtPointer, XtPointer)
{
   if (dialogForm)
   {
      XtUnmanageChild(dialogForm);
      return;
   }
}

MohrRockInfoClass::initialize(hook)
{
   MohrOptionType
      * parentAttrib = mohrOptionObj.getAttributes();
   XmString titleStr;
   char text[50];

   String text[] = {
      "Rock Type data obtained from\n"
      "   Brown, E. T. & Hoek, E.  (1980) Underground Excavations in Rock.",
      "      The Institution of Mining and Metallurgy, London.", NULL
   };

   if (dialogForm)
   {
      XtManageChild(dialogForm);
      return;
   }

   titleStr = XmStringCreateLocalized("Rock Data Citation");

   dialogForm = XmCreateFormDialog(hook, "Rock Data Citation",
                XmNdeleteResponse, XmUNMAP,
                XmNnoResize, TRUE,
                XmNdialogTitle, titleStr, NULL);

   Widget rowCol = XtVaCreateWidget("rowcol",
                   xmRowColumnWidgetClass, dialogForm, NULL);

   textArea = XmCreateScrolledText("textArea",
              xmTextWidgetClass, rowCol,
              XmNrows, 12,
              XmNcolumns, 70,
              XmNeditable, False,
              XmNeditMode, XmNMULTI_LINE_EDIT,
              XmNcursorPositionVisible, False,
              XmNwordWrap, True,
              XmNvalue, text, NULL);

   seperator = XtVaCreateManagedWidget("Seperator1",
               xmSeparatorGadgetClass, rowCol, NULL);

   closeButton = XtVaCreateManagedWidget("closeButton",
                 xmPushButtonWidgetClass, rowCol,
                 XtVaTypedArg, XmNlabelString,
                 XmRString, "Close", 10,
                 XmNbackground, PUSH_COLOR, NULL);

   XtAddCallback (closeButton, XmNactivateCallback,
                  functionCloseButton, NULL);

   XtManageChild(rowCol);
   XtManageChild(dialogForm);
}
```

mohrClass.hh

Added holder for effective stress to the attributes structure.

```
double effStress[3]; // effective stress (sigmas modified by fluid pressure)
```

mohrClass.c++

Added initialization of effStress[3] to the constructor.

```
attributes.effStress[0] = 0;      // sigma modified by fluid pressure
attributes.effStress[1] = 50;
attributes.effStress[2] = 100;
```

mohrOptionCB.c++

Changed updateEffective to handle updating effSig and dependent stresses

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  updateEffective (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  void
//
//  State Changes:  mohrObj.attributes->effStress are changed and this change
//                  is displayed on screen in the text fields
//
//  Purpose:  This function takes care of calculating effective stress
//
// Last Modified:  27 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void updateEffective()
{
   MohrOptionType  // option window attributes
      *moattr = mohrOptionObj.getAttributes();
   MohrAttributeType  // mohr graph attributes
      *mattr = mohrObj.getAttributes();

   char strs[3][100]; // holder for transfer of numbers to ASCII

   int
      i,    // loop index
      min = 0, mid = 0, max = 0; // hold indeces of smallest, middle, and largest
sigmas

   double
      ratio;   // ratio of delta sigmas[max] to delta sigmas[min]

   if (mattr.stressMode == 0) // stresses are dependent
   {
      ratio = mattr.stressDepRatio / (1 - mattr.stressDepRatio); // set up ratio a
ccording to formula

      for(i = 1; i < 3; i++)  // find max and min
      {
         if (mattr->sigmas[i] > mattr->sigmas[max]) // new max
            max = i;
         if (mattr->sigmas[i] < mattr->sigmas[max]) // new min
            min = i;
      } // end for and i

      for (i = 0; i < 3; i++)  // find the mid
      {
         if( (i != max) && (i != min) ) // mid is not min or max
            mid = i;
      } // end for and i

      // we are now ready to set the effective stresses
      mattr->effStress[max] = (double)mattr->sigmas[max] - mohrObj.fluidPressure;
      mattr->effStress[mid] = (double)mattr->sigmas[mid] - mohrObj.fluidPressure;
```



```
      mattr->effStress[min] = (double)mattr->sigmas[min] - (ratio * mohrObj.fluidP
ressure);
   } // end if (mattr.stressMode == 0)

   else // stresses are independent so set them strait
   {
      mattr->effStress[0] = (double)mattr->sigmas[0] - mohrObj.fluidPressure;
      mattr->effStress[1] = (double)mattr->sigmas[1] - mohrObj.fluidPressure;
      mattr->effStress[2] = (double)mattr->sigmas[2] - mohrObj.fluidPressure;
   } // end else

   for (i = 0; i < 3; i++)  // set the scales on the screen
   {
      sprintf(strs[i],"%5lf",mattr->effStress[i]);
      XmTextFieldSetString(moattr->effText[i], strs[i]);
   } // end for int i
} // end of updateEffective
```

Changed these functions to make use of effStress variables

```
void mohrApplyOKCB(Widget w, XtPointer, XtPointer)
void mohrApplyButton(Widget, XtPointer, XtPointer)
```

mohrClass.c++

Changed void MohrClass::failure(void) to make use of effStress variables

mohrCallbacks

Changed void mohrCBdraw() to make use of effStress variables (lines 250 - 252)

```
sigmas[0] = mattr->effStress[0];
sigmas[1] = mattr->effStress[1];
sigmas[2] = mattr->effStress[2];
```

mohrClass.hh

Changed int stressDepRatio in attributes structure

```
double stressDepRatio; // the degree to which stresses are related
```

mohrClass.c++

Changed attributes.stressScaleMin = 1; // Min Poisson's Ratio is 1%
to attributes.stressScaleMin = 2; // Min Poisson's Ratio is 2% in
initialization.

Also changed attributes.stressScaleMax = 100; // Max Poisson's Ratio is 100%
to attributes.stressScaleMax = 99; // Max Poisson's Ratio is 99% in
initialization

mohrOptionCB

Changed the following function to fix a bug dealing with Poisson's
ratio:

```
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
void mohrStressScaleChange(Widget, XtPointer clientData, XtPointer callData)
```

mohrOptionClass.hh

Changed stressScale to s3Tos1RatioScale and added s2Tos1RatioScale Widget to the attributes.

```
Widget   s3Tos1RatioScale; //  Scale for specifiying Poisson's

Widget   s2Tos1RatioScale; //  Scale for specifiying Poisson's
```

mohrOptionClass.c++

Added a more meaningful label to the Stress Ratio Scale and removed the old one. Also, added another scale for a stress ratio between s1 and s2 and the appropriate callback. Changed all occurances of stressScale to s3Tos1RatioScale.

```
// define a scale for the ratio between dependent stresses
XmString
   strOne, strTwo, strFinal;

strOne = motifFontObj.create("", "s", "3");
strTwo = motifFontObj.create(" to ", "s", "1 Ratio");
strFinal = XmStringConcat(strOne, strTwo);
XmStringFree(strOne);
XmStringFree(strTwo);
attributes.s3Tos1RatioScale = XtVaCreateManagedWidget("W",
                        xmScaleWidgetClass, col3,
                        XmNtitleString, strFinal,
                        XmNfontList, motifFontObj.fontListing,
                        XmNorientation, XmHORIZONTAL,
                        XmNmaximum, mattr->stressScaleMax,
                        XmNminimum, mattr->stressScaleMin,
                        XmNvalue, 25,
                        XmNscaleMultiple, 1,
                        XmNdecimalPoints, 2,
                        XmNshowValue, True,
                        XmNbackground, WIDGET_COLOR,  NULL);
XmStringFree(strFinal);

// Poisson's % scale should initially be inactive
XtSetSensitive(attributes.s3Tos1RatioScale, False);
// end of define a scale for the ratio between dependent stresses

strOne = motifFontObj.create("", "s", "2");
strTwo = motifFontObj.create(" to ", "s", "1 Ratio");
strFinal = XmStringConcat(strOne, strTwo);
XmStringFree(strOne);
XmStringFree(strTwo);
attributes.s2Tos1RatioScale = XtVaCreateManagedWidget("W",
                        xmScaleWidgetClass, col3,
                        XmNtitleString, strFinal,
                        XmNfontList, motifFontObj.fontListing,
                        XmNorientation, XmHORIZONTAL,
                        XmNmaximum, mattr->stressScaleMax,
                        XmNminimum, mattr->stressScaleMin,
                        XmNvalue, 25,
                        XmNscaleMultiple, 1,
                        XmNdecimalPoints, 2,
                        XmNshowValue, True,
                        XmNbackground, WIDGET_COLOR,  NULL);
XmStringFree(strFinal);

// Poisson's % scale should initially be inactive
XtSetSensitive(attributes.s2Tos1RatioScale, False);
// end of define a scale for the ratio between dependent stresses

XtAddCallback (attributes.s2Tos1RatioScale, XmNdragCallback,
   mohrs2Tos1RatioScaleChange, (XtPointer)0);
```

mohrClass.hh


```
   double s3Tos1Ratio; // the degree to which stresses are related
   double s2Tos1Ratio; // the degree to which stresses are related
```

mohrClass.c++

```
attributes.s3Tos1Ratio = .25; // ratios initially 25%
attributes.s2Tos1Ratio = .25; // ratios initially 25%
```

mohrOptionCB.hh

Added three new functions to complexity of the mohrScaleChange function when in dependent stress mode. Renamed mohrStressScaleChange to mohrs3Tos1RatioScaleChange and added mohrs2Tos1RatioScaleChange.

```
void changeMaxStress(int changeOthers, int max, int mid, int min, double delta, do
uble ratio);
void changeMidStress(int changeOthers, int max, int mid, int min, double delta, do
uble ratio);
void changeMinStress(int changeOthers, int max, int mid, int min, double delta, do
uble ratio);
void mohrs3Tos1RatioScaleChange(Widget, XtPointer, XtPointer);
void mohrs2Tos1RatioScaleChange(Widget, XtPointer, XtPointer);
```

mohrOptionCB.c++

Added the following three functions to cut complexity of the mohrScaleChange function when in dependent stress mode.

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMaxStress (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  int changeOthers -- 0 don't call changeMidStress or
//                                  changeMinstress;  1 call them
//              int max -- index of largest valued sigma
//              int mid,-- index of middle valued sigma
//              int min -- index of smallest valued sigma
//              double delta -- change in value
//              double ratio -- ratio to multiply delta by
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to delta and ratio values
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses are dependant upon eachother.  It is called from
//            mohrScaleChange or one of the other changeM??Stress
//            functions
//
// Last Modified:  30 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMaxStress(int changeOthers, int max, int mid, int min, double delta, do
uble ratio)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   double newValue = mohrAttrib->sigmas[max] + ratio * delta;

   // check the bounds before assigning newValue to sigma[max]
   if(newValue > mohrAttrib->maxScale)
   {  // if too big
      newValue = mohrAttrib->maxScale;   // set to upper bound
      changeOthers = 1; // this change affects other stresses
   }
   else if(newValue < mohrAttrib->minScale)
   {  // if too small
      newValue = mohrAttrib->minScale; // set to lower bound
      changeOthers = 1; // this change affects other stresses
   }
```

```
   if (changeOthers) // if the other stress should be updated
   {  // update delta after bounds check
      delta = newValue - mohrAttrib->sigmas[max];
      // change the other stresses
      changeMidStress(0, max, mid, min, delta, mohrAttrib->s2Tos1Ratio / (1 - mohr
Attrib->s2Tos1Ratio));
      changeMinStress(0, max, mid, min, delta, mohrAttrib->s3Tos1Ratio / (1 - mohr
Attrib->s3Tos1Ratio));
   }
   // save the new value
   mohrAttrib->sigmas[max] = newValue;
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMidStress (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  int changeOthers -- 0 don't call changeMaxStress;
//                                  1 call changeMaxStress
//              int max -- index of largest valued sigma
//              int mid,-- index of middle valued sigma
//              int min -- index of smallest valued sigma
//              double delta -- change in value
//              double ratio -- ratio to multiply delta by
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to delta and ratio values
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses are dependant upon eachother.  It is called from
//            mohrScaleChange or one of the other changeM??Stress
//            functions
//
// Last Modified:  30 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMidStress(int changeOthers, int max, int mid, int min, double delta, do
uble ratio)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   double newValue = mohrAttrib->sigmas[mid] + ratio * delta;

   // check the bounds before assigning newValue to sigma[mid]
   if(newValue > mohrAttrib->maxScale)
   {  // if too big
      newValue = mohrAttrib->maxScale;   // set to upper bound
      changeOthers = 1; // this change affects other stresses
   }
   else if(newValue < mohrAttrib->minScale)
   {  // if too small
      newValue = mohrAttrib->minScale; // set to lower bound
      changeOthers = 1; // this change affects other stresses
   }

   if (changeOthers) // if the other stress should be updated
   {  // update delta after bounds check
      delta = newValue - mohrAttrib->sigmas[mid];
      // change the other stresses
      changeMaxStress(1, max, mid, min, delta, (1 - mohrAttrib->s2Tos1Ratio) / moh
rAttrib->s2Tos1Ratio);
   }
   // save the new value
   mohrAttrib->sigmas[mid] = newValue;
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMinStress (not a member function)
//
//  File:  mohrOptionCB.c++
```



```
//
//  Arguments:  int changeOthers -- 0 don't call changeMaxStress;
//                                  1 call changeMaxStress
//              int max -- index of largest valued sigma
//              int mid -- index of middle valued sigma
//              int min -- index of smallest valued sigma
//              double delta -- change in value
//              double ratio -- ratio to multiply delta by
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to delta and ratio values
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses are dependant upon eachother.  It is called from
//            mohrScaleChange or one of the other changeM??Stress
//            functions
//
// Last Modified:  30 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMinStress(int changeOthers, int max, int mid, int min, double delta, do
uble ratio)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   double newValue = mohrAttrib->sigmas[min] + ratio * delta;

   // check the bounds before assigning newValue to sigma[min]
   if(newValue > mohrAttrib->maxScale)
   {  // if too big
      newValue = mohrAttrib->maxScale;   // set to upper bound
      changeOthers = 1; // this change affects other stresses
   }
   else if(newValue < mohrAttrib->minScale)
   {  // if too small
      newValue = mohrAttrib->minScale; // set to lower bound
      changeOthers = 1; // this change affects other stresses
   }

   if (changeOthers) // if the other stress should be updated
   {  // update delta after bounds check
      delta = newValue - mohrAttrib->sigmas[min];
      // change the other stresses
      changeMaxStress(1, max, mid, min, delta, (1 - mohrAttrib->s3Tos1Ratio) / moh
rAttrib->s3Tos1Ratio);
   }
   // save the new value
   mohrAttrib->sigmas[min] = newValue;
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              Ask Robert Boenau what the hell these are.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigma
//                  that was changed in the Mohr graph options window
//                  by the user.  Scales in the Options window are
//                  changed.
//
//  Purpose:  This program updates the sigmas according to the settings
//            of the scales in the Mohr graph options window devoted to
//            the sigmas.
//
// Last Modified:  16 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
```

```
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   //  The variable below holds the member variables of the
   //  class responsible for the Mohr Graph's Options window
   MohrOptionType *optionsAttrib = mohrOptionObj.getAttributes();

   //  The variable below, whichSig, is the index of the sigma that
   //  was changed in the Mohr's options window.  Also, newValue holds
   //  the value that the user set to the sigma indicated by whichSig,
   //  and delta holds the change in the sigma value.
   int
      whichSig = (int) clientData,
      max = 0, mid = 0, min = 0,  // indecies of largest, middle, & smallest sigma
values
      i;  // loop index

   double
      newValue = (double) cbs->value / 1000,
      delta = newValue - mohrObjAttrib->sigmas[whichSig];

   if(mohrObjAttrib->stressMode == 0)
   {  //stresses should depend on each other
      // Search for the largest and smallest sigmas
      for(i = 1; i < 3; i++)
      {
         if (mohrObjAttrib->sigmas[i] >= mohrObjAttrib->sigmas[max])
            max = i;
         if (mohrObjAttrib->sigmas[i] <= mohrObjAttrib->sigmas[min])
            min = i;
      }

      for(i = 0; i < 3; i++)
      {  // find mid, should be index not min and not max
         if ( (i != max) && (i != min) )
            mid = i;
      }

      if ( whichSig == max )
         changeMaxStress(1, max, mid, min, delta, 1);

      else if ( whichSig == min )
         changeMinStress(1, max, mid, min, delta, 1);

      else  // middle value changed
         changeMidStress(1, max, mid, min, delta, 1);

      XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max] * 1000
);
      XmScaleSetValue(optionsAttrib->scale[mid], mohrObjAttrib->sigmas[mid] * 1000
);
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min] * 1000
);
   }// end if(mohrObjAttrib->stressMode == 0)

   else  // scales should be inDependent
   {
      mohrObjAttrib->sigmas[whichSig] = newValue;
   }  // end else

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // end of mohrScaleChange

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```


Printed by jbuckner from yosemite


```
// Function:  mohrs3Tos1RatioScaleChange (not a member function)
//
// File:  mohrOptionCB.c++
//
// Arguments:  Widget, XtPointer, XtPointer callData
//
// State Changes:  mohrObj.attributes->s3Tos1Ratio, updates the degree
//                 to which the stress are dependent when in dependent
//                 stress mode
//
// Purpose:  This program updates the allows the user to change Poisson's
//           ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  30 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrs3Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   // update the ratio
   mohrObjAttrib->s3Tos1Ratio =  (double) (cbs->value / 100.0);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrs3Tos1RatioScaleChange

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrs2Tos1RatioScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.attributes->s2Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  30 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrs2Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   // update the ratio
   mohrObjAttrib->s2Tos1Ratio =  (double) (cbs->value / 100.0);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrs2Tos1RatioScaleChange

void mohrChangeMinMax(Widget, XtPointer clientData, XtPointer){
   MohrAttributeType *mattr = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
   int tmpValue;
```

```
if ((int)clientData == 1) { // change min
   if (XmTextFieldGetString(moattr->minText) != NULL) {
      char *val = XmTextFieldGetString(moattr->minText);
      if (strlen(val) == 0) {
         infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
         return;
      }
      tmpValue = atoi(val);
      if (tmpValue < -1000)
         tmpValue = -1000;
      mattr->minScale = tmpValue;
   } // end if
} // end if

else { // changed max
   if (XmTextFieldGetString(moattr->maxText) != NULL) {
      char *val = XmTextFieldGetString(moattr->maxText);
      if (strlen(val) == 0) {
         infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
         return;
      }
      tmpValue = atoi(val);
      if (tmpValue > 1000)
         tmpValue = 1000;
      mattr->maxScale = tmpValue;
   } // end if
} // end else

if (mattr->minScale >= mattr->maxScale){
   infoWidget(moattr->mohrDialogForm,
              "Min Scale must be less than Max Scale");
   return;
}
// Change value of the sliders and make sure present values lie
// within the min and max
for (int i = 0; i < 3; i++) {
   if (mattr->sigmas[i] < mattr->minScale ||
       mattr->sigmas[i] > mattr->maxScale) {
      mattr->sigmas[i] = mattr->minScale;
      XmScaleSetValue(moattr->scale[i], mattr->sigmas[i]);
   } // end if
   XtVaSetValues(moattr->scale[i],
                 XmNmaximum, mattr->maxScale,
                 XmNminimum, mattr->minScale,
                 NULL);
} // end for int i

mohrObj.display();
} // end of mohrChangeMinMax
```


All of

```
mohrOptionCB.c++:820:void mohrChangeFluidPress(Widget, XtPointer, XtPointer){
mohrOptionCB.c++:841:} // end of mohrChangeFluidPress
mohrOptionCB.hh:56:void mohrChangeFluidPress(Widget, XtPointer, XtPointer);
mohrOptionClass.c++:974:                    mohrChangeFluidPress, NULL);
```

changed to

```
mohrOptionCB.c++:820:void mohrChangeFluidField(Widget, XtPointer, XtPointer){
mohrOptionCB.c++:841:} // end of mohrChangeFluidField
mohrOptionCB.hh:56:void mohrChangeFluidField(Widget, XtPointer, XtPointer);
mohrOptionClass.c++:974:                    mohrChangeFluidField, NULL);
```

mohrOptionCB.hh

Added void mohrChangeFluidScale(Widget, XtPointer, XtPointer);

mohrOptionCB.c++

Added the following function:

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChangeFluidScale (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.fluidPressure, updates the fluidPressure
//                  also calls updateEffective(), mohrObj.failure(),
//                  mohrObj.display()
//
//  Purpose:  This program updates the fluid pressure from its scale
//            in the options window of the Mohr graph
//
// Last Modified:  1 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChangeFluidScale(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();

   char strHolder[100]; // holder for transfer of numbers to ASCII

   // update the ratio
   mohrObj.fluidPressure =  (double) (cbs->value);

   sprintf(strHolder,"%5lf", mohrObj.fluidPressure);
   XmTextFieldSetString(moattr->fluidText, strHolder);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrs3Tos1RatioScaleChange
```

Changed the following function:

```
void mohrChangeFluidField(Widget, XtPointer, XtPointer){
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
   MohrAttributeType * mohrObjAttrib = mohrObj.getAttributes();
   char strHolder[100]; // holder for transfer of numbers to ASCII
   double tmpValue;

   if (XmTextFieldGetString(moattr->fluidText) != NULL) {
```

Printed by jbuckner from yosemite


```
      char *val = XmTextFieldGetString(moattr->fluidText);
      if (strlen(val) == 0) {
         infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
         return;
      }
      tmpValue = atof(val);
      if (tmpValue < 0.0) {
         infoWidget(moattr->mohrDialogForm,"Fluid Pressure must be a positive numb
er");
         sprintf(strHolder,"%5lf", mohrObj.fluidPressure);
         XmTextFieldSetString(moattr->fluidText, strHolder);
         return;
      }

      if (tmpValue > mohrObjAttrib->maxScale){
         infoWidget(moattr->mohrDialogForm,"Fluid Pressure is out of bounds");
         sprintf(strHolder,"%5lf", mohrObj.fluidPressure);
         XmTextFieldSetString(moattr->fluidText, strHolder);
         return;
      }

      mohrObj.fluidPressure = tmpValue;

      XmScaleSetValue(moattr->fluidScale, mohrObj.fluidPressure);

      updateEffective();
      mohrObj.failure();
      mohrObj.display();
   } // end if

} // end of mohrChangeFluidField
```

mohrOptionClass.hh

Added Widget fluidScale to MohrOptionType struct.

mohrOptionClass.c++

Added scale for changing fluid pressure and the appropriate callback:

```
attributes.fluidScale = XtVaCreateManagedWidget("fluidScale",
                           xmScaleWidgetClass, col3,
                           XtVaTypedArg,
                           XmNtitleString, XmRString,
                           "Fluid Pressure", 14,
                           XmNorientation, XmHORIZONTAL,
                           XmNmaximum, mattr->maxScale,
                           XmNminimum, 0,
                           XmNvalue, 0,
                           XmNscaleMultiple, 1,
                           XmNshowValue, True,
                           XmNbackground, WIDGET_COLOR,  NULL);

XtAddCallback (attributes.fluidScale, XmNdragCallback,
               mohrChangeFluidScale, NULL);

Rearranged Widgets to make the window smaller:

Widget subColRow1  = XtVaCreateWidget("colWidget",
                        xmRowColumnWidgetClass,
                        row1,
                        XmNorientation, XmVERTICAL,
                        XmNnumColumns, 14,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

// define radio buttons for setting dependentcy of stresses on each other
frame = XtVaCreateManagedWidget("frameStress", xmFrameWidgetClass,
                                subColRow1,
                                XmNbackground, WIDGET_COLOR,
```



```
                                NULL);

   Widget colStress = XtVaCreateWidget("colStressWidget",
                           xmRowColumnWidgetClass, frame,
                           XmNorientation, XmVERTICAL,
                           XmNnumColumns, 7,
                           XmNpacking, XmPACK_TIGHT,
                           XmNbackground, WIDGET_COLOR,
                           NULL);

   Widget stressdependenceLabel = XtVaCreateManagedWidget("stressLable",
                           xmLabelWidgetClass, colStress,
                           XtVaTypedArg,
                           XmNlabelString, XmRString,
                           "Stress Mode", 12,
                           XmNalignment, XmALIGNMENT_CENTER,
                           XmNbackground, WIDGET_COLOR,
                           NULL);

   Widget seperatorStress = XtVaCreateManagedWidget("SeperatorStress",
                                    xmSeparatorGadgetClass,
                                    colStress, NULL);

   attributes.stressDependenceRadio = XmCreateRadioBox(colStress, "stressDependenc
eRadio", NULL, 0);
   XmChangeColor(attributes.stressDependenceRadio, WIDGET_COLOR);

   attributes.stressDependentRadio = XtVaCreateManagedWidget("stressDependentRadio
",
                                    xmToggleButtonGadgetClass,
                                    attributes.stressDependenceRadio,
                                    XtVaTypedArg,
                                    XmNlabelString, XmRString,
                                    "Dependent Stresses", 19,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

   attributes.stressIndependentRadio = XtVaCreateManagedWidget("stressIndependentR
adio",
                                    xmToggleButtonGadgetClass,
                                    attributes.stressDependenceRadio,
                                    XtVaTypedArg,
                                    XmNlabelString, XmRString,
                                    "Independent Stresses", 21,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

   if (mattr->stressMode == 1)
      XmToggleButtonSetState(attributes.stressIndependentRadio, TRUE, TRUE);
   else
      XmToggleButtonSetState(attributes.stressDependenceRadio, TRUE, TRUE);
   // end of section for stress dependency radio buttons

   // define a scale for the ratio between dependent stresses
   XmString
      strOne, strTwo, strFinal;

   strOne = motifFontObj.create("", "s", "3");
   strTwo = motifFontObj.create(" to ", "s", "1 Ratio");
   strFinal = XmStringConcat(strOne, strTwo);
   XmStringFree(strOne);
   XmStringFree(strTwo);
   attributes.s3TolsRatioScale = XtVaCreateManagedWidget("W",
                                    xmScaleWidgetClass, subColRow1,
                                    XmNtitleString, strFinal,
                                    XmNfontList, motifFontObj.fontListing,
                                    XmNorientation, XmHORIZONTAL,
                                    XmNmaximum, mattr->stressScaleMax,
                                    XmNminimum, mattr->stressScaleMin,
                                    XmNvalue, 25,
                                    XmNscaleMultiple, 1,
```

```
                            XmNdecimalPoints, 2,
                            XmNshowValue, True,
                            XmNbackground, WIDGET_COLOR,  NULL);
XmStringFree(strFinal);

// Poisson's % scale should initially be inactive
XtSetSensitive(attributes.s3Tos1RatioScale, False);
// end of define a scale for the ratio between dependent stresses

strOne = motifFontObj.create("", "s", "2");
strTwo = motifFontObj.create(" to ", "s", "1 Ratio");
strFinal = XmStringConcat(strOne, strTwo);
XmStringFree(strOne);
XmStringFree(strTwo);
attributes.s2Tos1RatioScale = XtVaCreateManagedWidget("W",
                            xmScaleWidgetClass, subColRow1,
                            XmNtitleString, strFinal,
                            XmNfontList, motifFontObj.fontListing,
                            XmNorientation, XmHORIZONTAL,
                            XmNmaximum, mattr->stressScaleMax,
                            XmNminimum, mattr->stressScaleMin,
                            XmNvalue, 25,
                            XmNscaleMultiple, 1,
                            XmNdecimalPoints, 2,
                            XmNshowValue, True,
                            XmNbackground, WIDGET_COLOR,  NULL);
XmStringFree(strFinal);

// Poisson's % scale should initially be inactive
XtSetSensitive(attributes.s2Tos1RatioScale, False);
// end of define a scale for the ratio between dependent stresses
```


mohrOptionClass.hh

Added two widgets for textual entry of dependent stress ratios:

```
Widget   s3Tos1RatioText,              //  Text box for entry of ratio
         s3Tos1RatioScale,             //  Scale for specifiying Poisson's
         s2Tos1RatioText,              //  Text box for entry of ratio
         s2Tos1RatioScale;             //  Scale for specifiying Poisson's
```

mohrOptionClass.c++

Added initialization for s3Tos1RatioText and s2Tos1RatioText:

```
XmString  // These are variables for producing the sigma labels
   strOne, strTwo, strFinal;

char strHolder[100]; // holder to initialize text box with number

strOne = motifFontObj.create("", "s", "3");  // sigma 3
strTwo = motifFontObj.create(" to ", "s", "1 Ratio"); // to sigma 1 ratio
strFinal = XmStringConcat(strOne, strTwo); // combine above two strings
XmStringFree(strOne);   // free first string

// lable for s3Tos1ratio text box and scale
Widget s3Tos1RatioLabel = XtVaCreateManagedWidget("s3Tos1RatioLabel",
                        xmLabelWidgetClass, subColRow1,
                        XmNlabelString, strFinal,
                        XmNfontList, motifFontObj.fontListing,
                        XmNbackground, WIDGET_COLOR,
                        NULL);
// free the title string
XmStringFree(strFinal);

// create the text box
attributes.s3Tos1RatioText = XtVaCreateManagedWidget("ratio1Text",
                        xmTextFieldWidgetClass,
                        subColRow1,
                        XmNtraversalOn, True,
                        XmNcolumns, 10,
                        XmNbackground, FILL_COLOR,
                        NULL);
// initialize the text box's value
sprintf(strHolder, "%lf", mattr->s3Tos1Ratio);
XmTextSetString(attributes.s3Tos1RatioText, strHolder);

// Poisson's % text box should initially be inactive
XtSetSensitive(attributes.s3Tos1RatioText, False);

// define a scale for the ratio between dependent stresses
attributes.s3Tos1RatioScale = XtVaCreateManagedWidget("ratio1Scale",
                         xmScaleWidgetClass, subColRow1,
                         XmNorientation, XmHORIZONTAL,
                         XmNmaximum, mattr->stressScaleMax,
                         XmNminimum, mattr->stressScaleMin,
                         XmNvalue, (int) (mattr->s3Tos1Ratio * 100),
                         XmNscaleMultiple, 1,
                         XmNdecimalPoints, 2,
                         XmNshowValue, False,
                         XmNbackground, WIDGET_COLOR,  NULL);

// Poisson's % scale should initially be inactive
XtSetSensitive(attributes.s3Tos1RatioScale, False);

strOne = motifFontObj.create("", "s", "2"); // sigma 2
strFinal = XmStringConcat(strOne, strTwo);  // concatenate two strings
XmStringFree(strOne);   // free the two extra stings, holders
XmStringFree(strTwo);

// lable for s2Tos1ratio text box and scale
```

```
Widget s2Tos1RatioLabel = XtVaCreateManagedWidget("s2Tos1RatioLabel",
                        xmLabelWidgetClass, subColRow1,
                        XmNlabelString, strFinal,
                        XmNfontList, motifFontObj.fontListing,
                        XmNbackground, WIDGET_COLOR,
                        NULL);
// free the title string
XmStringFree(strFinal);

// create a text box
attributes.s2Tos1RatioText = XtVaCreateManagedWidget("ratio2Text",
                        xmTextFieldWidgetClass,
                        subColRow1,
                        XmNtraversalOn, True,
                        XmNcolumns, 10,
                        XmNbackground, FILL_COLOR,
                        NULL);

// initialize the new text box with a number
sprintf(strHolder, "%lf", mattr->s2Tos1Ratio);
XmTextSetString(attributes.s2Tos1RatioText, strHolder);

// Poisson's % text should initially be inactive
XtSetSensitive(attributes.s2Tos1RatioText, False);

attributes.s2Tos1RatioScale = XtVaCreateManagedWidget("ratio2Scale",
                        xmScaleWidgetClass, subColRow1,
                        XmNorientation, XmHORIZONTAL,
                        XmNmaximum, mattr->stressScaleMax,
                        XmNminimum, mattr->stressScaleMin,
                        XmNvalue, (int) (mattr->s2Tos1Ratio * 100),
                        XmNscaleMultiple, 1,
                        XmNdecimalPoints, 2,
                        XmNshowValue, False,
                        XmNbackground, WIDGET_COLOR,  NULL);

// Poisson's % scale should initially be inactive
XtSetSensitive(attributes.s2Tos1RatioScale, False);

// end of define a text and scale for dependent stresses ratios

XtAddCallback (attributes.s3Tos1RatioText, XmNactivateCallback,
               mohrChanges3Tos1RatioField, NULL);
XtAddCallback (attributes.s3Tos1RatioScale, XmNdragCallback,
               mohrs3Tos1RatioScaleChange, (XtPointer)0);

XtAddCallback (attributes.s2Tos1RatioText, XmNactivateCallback,
               mohrChanges2Tos1RatioField, NULL);
XtAddCallback (attributes.s2Tos1RatioScale, XmNdragCallback,
               mohrs2Tos1RatioScaleChange, (XtPointer)0);

Changed fluid text and scale to resemble stress ratio text and scale:

rangeLabel = XtVaCreateManagedWidget("fluidLabel",
                        xmLabelWidgetClass, col3,
                        XtVaTypedArg,
                        XmNlabelString, XmRString,
                        "Fluid Pressure", 14,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

attributes.fluidText = XtVaCreateManagedWidget("minText",
                        xmTextFieldWidgetClass,
                        col3,
                        XmNtraversalOn, True,
                        XmNcolumns, 10,
                        XmNbackground, FILL_COLOR,
                        NULL);

sprintf(strHolder, "%d", mohrObj.fluidPressure);
XmTextSetString(attributes.fluidText, strHolder);
```



```
   attributes.fluidScale = XtVaCreateManagedWidget("fluidScale",
                        xmScaleWidgetClass, col3,
                        XmNorientation, XmHORIZONTAL,
                        XmNmaximum, mattr->maxScale,
                        XmNminimum, 0,
                        XmNvalue, (int) mohrObj.fluidPressure,
                        XmNscaleMultiple, 1,
                        XmNshowValue, False,
                        XmNbackground, WIDGET_COLOR,  NULL);

mohrOptionCB.hh

   Added functions to handle stress ratio text boxes:

void mohrChanges3Tos1RatioField(Widget, XtPointer, XtPointer);
void mohrChanges2Tos1RatioField(Widget, XtPointer, XtPointer);

mohrOptionCB.c++

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChanges3Tos1RatioField (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  mohrObj.attributes->s3Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  2 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChanges3Tos1RatioField(Widget, XtPointer, XtPointer)
{
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();
   MohrAttributeType // get Mohr graph stats
      * mohrObjAttrib = mohrObj.getAttributes();

   char
      strHolder[100], // holder for transfer of numbers to ASCII
      * val = XmTextFieldGetString(moattr->s3Tos1RatioText); // get new value

   double tmpValue;  // holder for new value in numeric form

   if (strlen(val) == 0) // is field is completely blank
   {  // tell user of mistake
      infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
      sprintf(strHolder,"%5lf", mohrObjAttrib->s3Tos1Ratio);    // reset field
      XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
      return;
   }

   tmpValue = atof(val);  // convert new value to numeric form

   if ( (tmpValue > mohrObjAttrib->stressScaleMax) || (tmpValue < mohrObjAttrib->s
tressScaleMin) )
   {  // check that new value is within bounds
         infoWidget(moattr->mohrDialogForm,"Ratio out of bounds");  // dialog box
error message
         sprintf(strHolder,"%5lf", mohrObjAttrib->s3Tos1Ratio);  // reset field
         XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
         return;
   }

   mohrObjAttrib->s3Tos1Ratio = tmpValue; // update variable
   // update scale
```

```
   XmScaleSetValue(moattr->s3Tos1RatioScale, (int) (mohrObjAttrib->s3Tos1Ratio * 1
00));

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChanges2Tos1RatioField (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  mohrObj.attributes->s2Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  2 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChanges2Tos1RatioField(Widget, XtPointer, XtPointer)
{
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();
   MohrAttributeType // get Mohr graph stats
      * mohrObjAttrib = mohrObj.getAttributes();

   char
      strHolder[100], // holder for transfer of numbers to ASCII
      * val = XmTextFieldGetString(moattr->s2Tos1RatioText); // get new value

   double tmpValue;  // holder for new value in numeric form

   if (strlen(val) == 0) // is field is completely blank
   {  // tell user of mistake
      infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
      sprintf(strHolder,"%5lf", mohrObjAttrib->s2Tos1Ratio);     // reset field
      XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
      return;
   }

   tmpValue = atof(val);  // convert new value to numeric form

   if ( (tmpValue > mohrObjAttrib->stressScaleMax) || (tmpValue < mohrObjAttrib->s
tressScaleMin) )
   {  // check that new value is within bounds
         infoWidget(moattr->mohrDialogForm,"Ratio out of bounds");  // dialog box
error message
         sprintf(strHolder,"%5lf", mohrObjAttrib->s2Tos1Ratio);  // reset field
         XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
         return;
   }

   mohrObjAttrib->s2Tos1Ratio = tmpValue; // update variable
   // update scale
   XmScaleSetValue(moattr->s2Tos1RatioScale, (int) (mohrObjAttrib->s3Tos1Ratio * 1
00));

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrs3Tos1RatioScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
```



```
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.attributes->s3Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  30 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrs3Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   char
      strHolder[100]; // holder for transfer of numbers to ASCII

   // update the ratio
   mohrObjAttrib->s3Tos1Ratio =  (double) (cbs->value / 100.0);

   sprintf(strHolder,"%5lf", mohrObjAttrib->s3Tos1Ratio);  // reset field
   XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrs3Tos1RatioScaleChange

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrs2Tos1RatioScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.attributes->s2Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  30 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrs2Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   char
      strHolder[100]; // holder for transfer of numbers to ASCII
```

```
   // update the ratio
   mohrObjAttrib->s2Tos1Ratio =  (double) (cbs->value / 100.0);

   sprintf(strHolder,"%5lf", mohrObjAttrib->s2Tos1Ratio);  // reset field
   XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrs2Tos1RatioScaleChange
```

Modified void updateEffective() to be sure that effective stress is within bounds of graph:

```
   for (i = 0; i < 3; i++)  // set the scales on the screen
   {
      // check bounds
      if (mattr->effStress[i] > mattr->maxScale)
      {
         infoWidget(moattr->mohrDialogForm,"Warning:  Actual Effective Stress Valu
e out of bounds.\nSetting value to an artifical bound value.");
         mattr->effStress[i] = mattr->maxScale;
      }
      if (mattr->effStress[i] < mattr->minScale)
      {
         infoWidget(moattr->mohrDialogForm,"Warning:  Actual Effective Stress Valu
e out of bounds.\nSetting value to an artifical bound value.");
         mattr->effStress[i] = mattr->minScale;
      }
      sprintf(strs[i],"%5lf",mattr->effStress[i]);
      XmTextFieldSetString(moattr->effText[i], strs[i]);
   } // end for int i
```

mohrCallbacks.c++

line 409: corrected the spelling of "Failure has occurred"

mohrOptionClass.c++

Cleaned up the Option Window's GUI by adding a colum and frame and rearranging another frame:

```
   Widget col6  = XtVaCreateWidget("col6Widget",
                        xmRowColumnWidgetClass,
                        frame,
                        XmNorientation, XmVERTICAL,
                        XmNnumColumns, 20,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

   Widget row1  = XtVaCreateWidget("ro2Widget",
                        xmRowColumnWidgetClass,
                        col6,
                        XmNorientation, XmHORIZONTAL,
                        XmNnumColumns, 2,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

   frame = XtVaCreateManagedWidget("frame", xmFrameWidgetClass,
                                   col6,
                                   XmNbackground, WIDGET_COLOR, NULL);

   Widget subCol6  = XtVaCreateWidget("col6Widget",
                        xmRowColumnWidgetClass,
                        frame,
                        XmNorientation, XmVERTICAL,
                        XmNnumColumns, 10,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

   Widget effectiveLabel = XtVaCreateManagedWidget("effectiveLabel",
                        xmLabelWidgetClass, subCol6,
                        XtVaTypedArg,
                        XmNlabelString, XmRString,
                        "Effective Stress", 17,
                        XmNalignment, XmALIGNMENT_CENTER,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

   Widget row5 = XtVaCreateWidget("row5Widget",
                        xmRowColumnWidgetClass,
                        subCol6,
                        XmNorientation, XmHORIZONTAL,
                        XmNnumColumns, 2,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

   XmString effStr;
   effStr = motifFontObj.create("", "s", " U ");

   effectiveLabel = XtVaCreateManagedWidget("effectiveLabel",
                        xmLabelWidgetClass, row5,
                        XmNlabelString, effStr,
                        XmNfontList, motifFontObj.fontListing,
                        XmNalignment, XmALIGNMENT_CENTER,
                        XmNbackground, WIDGET_COLOR,
                        NULL);
   XmStringFree(effStr);

   attributes.effText[0] = XtVaCreateManagedWidget("effText",
                        xmTextFieldWidgetClass,
                        row5,
                        XmNtraversalOn, True,
                        XmNcolumns, 10,
```

```
                        XmNeditable, False,
                        XmNbackground, PUSH_COLOR,
                        NULL);

    Widget row6 = XtVaCreateWidget("row6Widget",
                        xmRowColumnWidgetClass,
                        subCol6,
                        XmNorientation, XmHORIZONTAL,
                        XmNnumColumns, 2,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

    effStr = motifFontObj.create("", "s", " V ");
    effectiveLabel = XtVaCreateManagedWidget("effectiveLabel",
                        xmLabelWidgetClass, row6,
                        XmNlabelString, effStr,
                        XmNfontList, motifFontObj.fontListing,
                        XmNalignment, XmALIGNMENT_CENTER,
                        XmNbackground, WIDGET_COLOR,
                        NULL);
    XmStringFree(effStr);

    attributes.effText[1] = XtVaCreateManagedWidget("effText",
                        xmTextFieldWidgetClass,
                        row6,
                        XmNtraversalOn, True,
                        XmNcolumns, 10,
                        XmNeditable, False,
                        XmNbackground, PUSH_COLOR,
                        NULL);

    Widget row7 = XtVaCreateWidget("row7Widget",
                        xmRowColumnWidgetClass,
                        subCol6,
                        XmNorientation, XmHORIZONTAL,
                        XmNnumColumns, 2,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);


    effStr = motifFontObj.create("", "s", " W ");
    effectiveLabel = XtVaCreateManagedWidget("effectiveLabel",
                        xmLabelWidgetClass, row7,
                        XmNlabelString, effStr,
                        XmNfontList, motifFontObj.fontListing,
                        XmNalignment, XmALIGNMENT_CENTER,
                        XmNbackground, WIDGET_COLOR,
                        NULL);
    XmStringFree(effStr);


    attributes.effText[2] = XtVaCreateManagedWidget("effText",
                        xmTextFieldWidgetClass,
                        row7,
                        XmNtraversalOn, True,
                        XmNcolumns, 10,
                        XmNeditable, False,
                        XmNbackground, PUSH_COLOR,
                        NULL);

    XmTextSetString(attributes.effText[0], "0");
    XmTextSetString(attributes.effText[1], "50");
    XmTextSetString(attributes.effText[2], "100");

=============================================================================

    frame = XtVaCreateManagedWidget("frameStress", xmFrameWidgetClass,
                                    subColRow1,
                                    XmNbackground, WIDGET_COLOR,
```

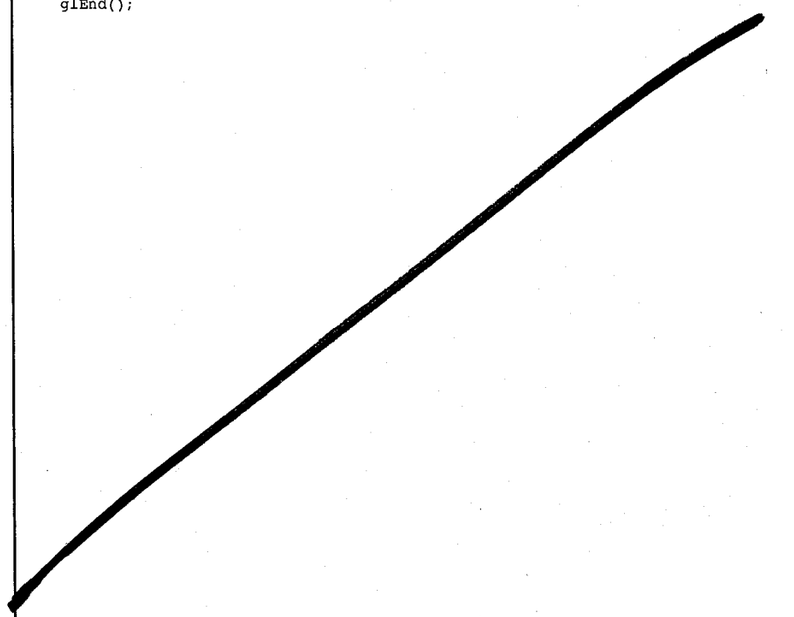


```
                                    NULL);

    Widget colStress = XtVaCreateWidget("colStressWidget",
                        xmRowColumnWidgetClass, frame,
                        XmNorientation, XmVERTICAL,
                        XmNnumColumns, 7,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

    Widget stressdependenceLabel = XtVaCreateManagedWidget("stressLable",
                        xmLabelWidgetClass, colStress,
                        XtVaTypedArg,
                        XmNlabelString, XmRString,
                        "Stress Mode", 12,
                        XmNalignment, XmALIGNMENT_CENTER,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

    Widget seperatorStress = XtVaCreateManagedWidget("SeperatorStress",
                                    xmSeparatorGadgetClass,
                                    colStress, NULL);

    attributes.stressDependenceRadio = XmCreateRadioBox(colStress, "stressDependenc
eRadio", NULL, 0);
    XmChangeColor(attributes.stressDependenceRadio, WIDGET_COLOR);

    attributes.stressDependentRadio = XtVaCreateManagedWidget("stressDependentRadio
",
                                    xmToggleButtonGadgetClass,
                                    attributes.stressDependenceRadio,
                                    XtVaTypedArg,
                                    XmNlabelString, XmRString,
                                    "Dependent Stresses", 19,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

    attributes.stressIndependentRadio = XtVaCreateManagedWidget("stressIndependentR
adio",
                                    xmToggleButtonGadgetClass,
                                    attributes.stressDependenceRadio,
                                    XtVaTypedArg,
                                    XmNlabelString, XmRString,
                                    "Independent Stresses", 21,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

    if (mattr->stressMode == 1)
        XmToggleButtonSetState(attributes.stressIndependentRadio, TRUE, TRUE);
    else
        XmToggleButtonSetState(attributes.stressDependenceRadio, TRUE, TRUE);
    // end of section for stress dependency radio buttons

    XmString  // These are variables for producing the sigma labels
        strOne, strTwo, strFinal;

    char strHolder[100]; // holder to initialize text box with number

    strOne = motifFontObj.create("", "s", "3");  // sigma 3
    strTwo = motifFontObj.create(" to ", "s", "1 Ratio"); // to sigma 1 ratio
    strFinal = XmStringConcat(strOne, strTwo); // combine above two strings
    XmStringFree(strOne);    // free first string

    // lable for s3Tos1ratio text box and scale
    Widget s3Tos1RatioLabel = XtVaCreateManagedWidget("s3Tos1RatioLabel",
                                xmLabelWidgetClass, colStress,
                                XmNlabelString, strFinal,
                                XmNfontList, motifFontObj.fontListing,
                                XmNbackground, WIDGET_COLOR,
                                NULL);
    // free the title string
```

```
    XmStringFree(strFinal);

    // create the text box
    attributes.s3Tos1RatioText = XtVaCreateManagedWidget("ratio1Text",
                            xmTextFieldWidgetClass,
                            colStress,
                            XmNtraversalOn, True,
                            XmNcolumns, 10,
                            XmNbackground, FILL_COLOR,
                            NULL);
    // initialize the text box's value
    sprintf(strHolder, "%lf", mattr->s3Tos1Ratio);
    XmTextSetString(attributes.s3Tos1RatioText, strHolder);

    // Poisson's % text box should initially be inactive
    XtSetSensitive(attributes.s3Tos1RatioText, False);

    // define a scale for the ratio between dependent stresses
    attributes.s3Tos1RatioScale = XtVaCreateManagedWidget("ratio1Scale",
                            xmScaleWidgetClass, colStress,
                            XmNorientation, XmHORIZONTAL,
                            XmNmaximum, mattr->stressScaleMax,
                            XmNminimum, mattr->stressScaleMin,
                            XmNvalue, (int) (mattr->s3Tos1Ratio * 100),
                            XmNscaleMultiple, 1,
                            XmNdecimalPoints, 2,
                            XmNshowValue, False,
                            XmNbackground, WIDGET_COLOR,  NULL);

    // Poisson's % scale should initially be inactive
    XtSetSensitive(attributes.s3Tos1RatioScale, False);

    strOne = motifFontObj.create("", "s", "2"); // sigma 2
    strFinal = XmStringConcat(strOne, strTwo);  // concatenate two strings
    XmStringFree(strOne);    // free the two extra stings, holders
    XmStringFree(strTwo);

    // lable for s2Tos1ratio text box and scale
    Widget s2Tos1RatioLabel = XtVaCreateManagedWidget("s2Tos1RatioLabel",
                            xmLabelWidgetClass, colStress,
                            XmNlabelString, strFinal,
                            XmNfontList, motifFontObj.fontListing,
                            XmNbackground, WIDGET_COLOR,
                            NULL);
    // free the title string
    XmStringFree(strFinal);

    // create a text box
    attributes.s2Tos1RatioText = XtVaCreateManagedWidget("ratio2Text",
                            xmTextFieldWidgetClass, colStress,
                            XmNtraversalOn, True,
                            XmNcolumns, 10,
                            XmNbackground, FILL_COLOR,
                            NULL);

    // initialize the new text box with a number
    sprintf(strHolder, "%lf", mattr->s2Tos1Ratio);
    XmTextSetString(attributes.s2Tos1RatioText, strHolder);

    // Poisson's % text should initially be inactive
    XtSetSensitive(attributes.s2Tos1RatioText, False);

    attributes.s2Tos1RatioScale = XtVaCreateManagedWidget("ratio2Scale",
                            xmScaleWidgetClass, colStress,
                            XmNorientation, XmHORIZONTAL,
                            XmNmaximum, mattr->stressScaleMax,
                            XmNminimum, mattr->stressScaleMin,
                            XmNvalue, (int) (mattr->s2Tos1Ratio * 100),
                            XmNscaleMultiple, 1,
                            XmNdecimalPoints, 2,
                            XmNshowValue, False,
```



```
                            XmNbackground, WIDGET_COLOR,  NULL);

    // Poisson's % scale should initially be inactive
    XtSetSensitive(attributes.s2Tos1RatioScale, False);

    // end of define a text and scale for dependent stresses ratios

mohrCallbacks.c++

    Fixed gap in fault envelope: (lines approx. 329 - 345)

    double tmpN = mattr->maxScale;
    glBegin(GL_LINE_STRIP);
       for (i = 1000+mattr->maxScale; i >= 0; i--) {
          if (!mohrObj.tauErrors[i]) {
             glVertex2d(tmpN, mohrObj.tauValues[i]);
          }
          tmpN -= 1.0;
       } // end for

       tmpN = -1000.0;

       for (i = 0; i < 1001+mattr->maxScale; i++) {
          if (!mohrObj.tauErrors[i]) {
             glVertex2d(tmpN, -mohrObj.tauValues[i]);
          }
          tmpN += 1.0;
       } // end for
    glEnd();
```



Jul 7 1997 16:35 **7July97.log** Page 1

Attempted to break the mohrOption window up into a main window with several child windows.

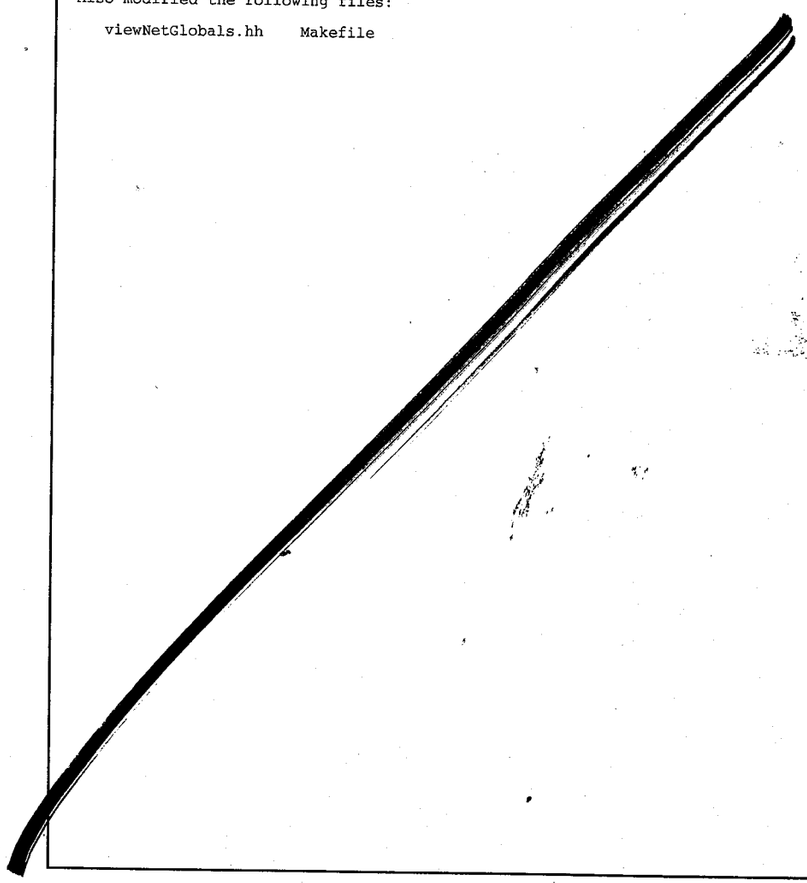
See the following files which were written on 7 July 1997:

mohrOpDisplayClass.c++	mohrOpRockClass.c++	mohrOpStressClass.c++
mohrOpDisplayClass.hh	mohrOpRockClass.hh	mohrOpStressClass.hh
mohrOptionClass.c++	mohrOptionObj.hh	mohrOpDisplayObj.hh
mohrOpRockObj.hh	mohrOpStressObj.hh	mohrOptionClass.hh

The previous convention of using a structure to hold attributes and returning a pointer to that structure for external function access was scrapped in favor of the use of friend functions for Motif callbacks.

Also modified the following files:

viewNetGlobals.hh Makefile



7July

Jul 9 1997 16:19 **9July97.log** Page 1

Fixed bug in

mohrOptionCB.c++

```
void mohrChangeMinMax(Widget, XtPointer clientData, XtPointer)

   for (int i = 0; i < 3; i++) {
      if (mattr->sigmas[i] < mattr->minScale ||
          mattr->sigmas[i] > mattr->maxScale) {
         mattr->sigmas[i] = mattr->minScale;
         XmScaleSetValue(moattr->scale[i], mattr->sigmas[i] * 1000);
      } // end if
      XtVaSetValues(moattr->scale[i],
                    XmNmaximum, mattr->maxScale * 1000,
                    XmNminimum, mattr->minScale * 1000,
                    NULL);
   } // end for int i
```

Toyed with a new method of calculating dependent stresses:

mohrOptionCB.hh:

Changed and added functions as follows:

```
void changeMaxStress(double newVal);
void changeMidStress(double newVal);
void changeMinStress(double newVal);
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts);
void checkStressBounds();
```

mohrOptionCB.c++

Changed and added functions as follows:

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMaxStress (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newVal -- value to store as new max sigma
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to newVal, mohrAttrib->s2Tos1Ratio, and
//                  mohrAttrib->s3Tos1Ratio
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses depend on each other.  It is called from
//            mohrScaleChange
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMaxStress(double newVal)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   int
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3);

   if (newVal > mohrAttrib->maxScale)
      newVal = mohrAttrib->stressScaleMax;
   if (newVal < mohrAttrib->minScale)
      newVal = mohrAttrib->stressScaleMin;

   mohrAttrib->sigmas[max] = newVal;

   if( (mohrAttrib->sigmas[mid] / newVal) > 0.0 )
      mohrAttrib->sigmas[mid] = mohrAttrib->s2Tos1Ratio * newVal;
   else
```

9July9

```
      mohrAttrib->sigmas[mid] = -1.0 * mohrAttrib->s2Tos1Ratio * newVal;

   if( (mohrAttrib->sigmas[min] / newVal) > 0.0 )
      mohrAttrib->sigmas[min] = mohrAttrib->s3Tos1Ratio * newVal;
   else
      mohrAttrib->sigmas[min] = -1.0 * mohrAttrib->s3Tos1Ratio * newVal;

   checkStressBounds();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMidStress (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newVal -- value to store as new mid sigma
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to newVal, mohrAttrib->s2Tos1Ratio, and
//                  mohrAttrib->s3Tos1Ratio
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses depend on each other.  It is called from
//            mohrScaleChange
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMidStress(double newVal)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   int
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3);

   if (newVal > mohrAttrib->maxScale)
      newVal = mohrAttrib->stressScaleMax;
   if (newVal < mohrAttrib->minScale)
      newVal = mohrAttrib->stressScaleMin;

   mohrAttrib->sigmas[mid] = newVal;

   if( (mohrAttrib->sigmas[max] / newVal) > 0.0 )
      mohrAttrib->sigmas[max] = (1.0 / mohrAttrib->s2Tos1Ratio) * newVal;
   else
      mohrAttrib->sigmas[max] = -1.0 * (1.0 / mohrAttrib->s2Tos1Ratio) * newVal;

   if( (mohrAttrib->sigmas[min] / newVal) > 0.0 )
      mohrAttrib->sigmas[min] = mohrAttrib->s3Tos1Ratio * mohrAttrib->sigmas[max];
   else
      mohrAttrib->sigmas[min] = -1.0 * mohrAttrib->s3Tos1Ratio * mohrAttrib->sigma
s[max];

   checkStressBounds();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMinStress (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newVal -- value to store as new max sigma
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to newVal, mohrAttrib->s2Tos1Ratio, and
//                  mohrAttrib->s3Tos1Ratio
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses depend on each other.  It is called from
```



```
//            mohrScaleChange
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMinStress(double newVal)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   int
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3);

   if (newVal > mohrAttrib->maxScale)
      newVal = mohrAttrib->stressScaleMax;
   if (newVal < mohrAttrib->minScale)
      newVal = mohrAttrib->stressScaleMin;

   mohrAttrib->sigmas[min] = newVal;

   if( (mohrAttrib->sigmas[max] / newVal) > 0.0 )
      mohrAttrib->sigmas[max] = (1.0 / mohrAttrib->s3Tos1Ratio) * newVal;
   else
      mohrAttrib->sigmas[max] = -1.0 * (1.0 / mohrAttrib->s3Tos1Ratio) * newVal;

   if( (mohrAttrib->sigmas[mid] / newVal) > 0.0 )
      mohrAttrib->sigmas[mid] = mohrAttrib->s2Tos1Ratio * mohrAttrib->sigmas[max];
   else
      mohrAttrib->sigmas[mid] = -1.0 * mohrAttrib->s2Tos1Ratio * mohrAttrib->sigma
s[max];

   checkStressBounds();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeStressBounds (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  none
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to bound violated, mohrAttrib->s2Tos1Ratio, and
//                  mohrAttrib->s3Tos1Ratio
//
//  Purpose:  This function is used to handle stresses that go beyond the
//            valid bounds when stresses depend on each other and are changed.
//            It is called from each of the changeM??Stress functions
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void checkStressBounds()
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   int
      changes[3] = {FALSE,FALSE,FALSE},  // to tell which sigs are changed
      i, // loop index
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3);

   for(i = 0; i < 3; i++)
   {
      if(mohrAttrib->sigmas[i] > mohrAttrib->maxScale)
      {
         changes[i] = TRUE;
         mohrAttrib->sigmas[i] = mohrAttrib->maxScale;
      }
```

```
      else if(mohrAttrib->sigmas[i] < mohrAttrib->minScale)
      {
         changes[i] = TRUE;
         mohrAttrib->sigmas[i] = mohrAttrib->minScale;
      }
   }

   if(changes[max])
   {
      mohrAttrib->sigmas[mid] = mohrAttrib->s2Tos1Ratio * mohrAttrib->sigmas[max];
      mohrAttrib->sigmas[min] = mohrAttrib->s3Tos1Ratio * mohrAttrib->sigmas[max];
   }
   else if(changes[mid])
   {
      mohrAttrib->sigmas[max] = (1.0 / mohrAttrib->s2Tos1Ratio) * mohrAttrib->sigm
as[mid];
      mohrAttrib->sigmas[min] = mohrAttrib->s3Tos1Ratio * mohrAttrib->sigmas[max];
   }
   else if(changes[min])
   {
      mohrAttrib->sigmas[max] = (1.0 / mohrAttrib->s3Tos1Ratio) * mohrAttrib->sigm
as[min];
      mohrAttrib->sigmas[mid] = mohrAttrib->s2Tos1Ratio * mohrAttrib->sigmas[max];
   }
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  findMinMidMax (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  int * min, int * mid, int * max, -- addresses to store info in
//              double values[] -- values to get min, mid, max indecies of
//              int noelts -- number of elements in values[]
//
//  State changes:  min, mid, and max are changed to reflect the indecies of
//                  minimum, middle, and maximum values in int values[]
//
//  Purpose:  This function is used to find the indecies of minimum, middle,
//            and maximum values in double values[]
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts)
{
   int i;   // loop index
   *min = *mid = *max = 0;       // initialize variables

   for(i = 0; i < noelts; i++) // find the max and min
   {
      if (values[i] > values[*max])
         *max = i; // new max has been found
      if (values[i] < values[*min])
         *min = i; // new min has been found
   }

   for(i = 0; i < noelts; i++)  // find mid
   {
      if( (i != *min) && (i != *max) )
         *mid = i; // mid has been found
   }
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigma
```



```
//                that was changed in the Mohr graph options window
//                by the user.  Scales in the Options window are
//                changed.
//
//  Purpose:  This program updates the sigmas according to the settings
//            of the scales in the Mohr graph options window devoted to
//            the sigmas.
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   //  The variable below holds the member variables of the
   //  class responsible for the Mohr Graph's Options window
   MohrOptionType *optionsAttrib = mohrOptionObj.getAttributes();

   //  The variable below, whichSig, is the index of the sigma that
   //  was changed in the Mohr's options window.  Also, newValue holds
   //  the value that the user set to the sigma indicated by whichSig,
   //  and delta holds the change in the sigma value.
   int
      whichSig = (int) clientData,
      min, mid, max;

   double
      newValue = (double) cbs->value / 1000;

   if(mohrObjAttrib->stressMode == 0)
   {  //stresses should depend on each other
      // find the index of the min, mid, and max sigmas
      findMinMidMax( & min, & mid, & max, mohrObjAttrib->sigmas, 3);

      if ( whichSig == max )  // max sigma was changed
         changeMaxStress(newValue);

      else if ( whichSig == min ) // min sigma was changed
         changeMinStress(newValue);

      else  // middle value changed
         changeMidStress(newValue);

      XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max] * 1000
);
      XmScaleSetValue(optionsAttrib->scale[mid], mohrObjAttrib->sigmas[mid] * 1000
);
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min] * 1000
);
   }// end if(mohrObjAttrib->stressMode == 0)

   else  // scales should be inDependent
   {
      mohrObjAttrib->sigmas[whichSig] = newValue;
   }  // end else

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // end of mohrScaleChange
```

Discovered major flaw in the philosophy of the dependent stress methods:

If fixed ratios are used to determine stresses, then the stresses (max, mid, and mid in the code) should never trade places with each other (as occurs with indepenent stresses).

Tried to come up with a useful solution to this problem:

One possible solution is to check after recalculating stresses with code such as:

in changeMaxStress function:
if sigmas[max] < sigmas[mid] then abort change

in changeMinStress function:
if sigmas[min] > sigmas[mid] then abort change

in changeMidStress function:
if sigmas[mid] > sigmas[max] or sigmas[mid] < sigmas[min], then abort change

Tinkered with various funtions in mohrOptionCB.c++. Much additional cleaning up is needed, but dependent stresses are almost reliable at this point.

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMaxStress (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newVal -- value to store as new max sigma
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to newVal, mohrAttrib->s2Tos1Ratio, and
//                  mohrAttrib->s3Tos1Ratio
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses depend on each other.  It is called from
//            mohrScaleChange
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMaxStress(double newVal)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   // tmpSigmas are sigmas right shifted to handle negatives as
   // lesser positives so that ratios work correctly
   double
      tmpSigmas[3] = {mohrAttrib->sigmas[0] + abs(mohrAttrib->minScale),
                      mohrAttrib->sigmas[1] + abs(mohrAttrib->minScale),
                      mohrAttrib->sigmas[2] + abs(mohrAttrib->minScale)};

   int
      min, mid, max; // indecies of min, mid, and max sigma values

   findMinMidMax( & min, & mid, & max, tmpSigmas, 3);

   // see than new value is not out of bounds
   if (newVal > mohrAttrib->maxScale)
      newVal = mohrAttrib->stressScaleMax;
   if (newVal < mohrAttrib->minScale)
      newVal = mohrAttrib->stressScaleMin;

   // right shift new value to match temp sigmas
   // newVal = newVal + abs(mohrAttrib->minScale);

   // figure the new values using the ratios
   tmpSigmas[max] = newVal + abs(mohrAttrib->minScale);
```



```
   tmpSigmas[mid] = (mohrAttrib->s2Tos1Ratio * newVal) + abs(mohrAttrib->minScale)
;
   tmpSigmas[min] = (mohrAttrib->s3Tos1Ratio * newVal) + abs(mohrAttrib->minScale)
;

   if (tmpSigmas[max] <= tmpSigmas[mid]) // if bad values
      return; // then abort change

   // correct sigmas are tmp sigmas left shifted
   mohrAttrib->sigmas[0] = tmpSigmas[0] - abs(mohrAttrib->minScale);
   mohrAttrib->sigmas[1] = tmpSigmas[1] - abs(mohrAttrib->minScale);
   mohrAttrib->sigmas[2] = tmpSigmas[2] - abs(mohrAttrib->minScale);

   // check the bounds on the new sigmas
   checkStressBounds();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMidStress (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newVal -- value to store as new mid sigma
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to newVal, mohrAttrib->s2Tos1Ratio, and
//                  mohrAttrib->s3Tos1Ratio
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses depend on each other.  It is called from
//            mohrScaleChange
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMidStress(double newVal)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   // tmpSigmas are sigmas right shifted to handle negatives as
   // lesser positives so that ratios work correctly
   double
      tmpSigmas[3] = {mohrAttrib->sigmas[0] + abs(mohrAttrib->minScale),
                      mohrAttrib->sigmas[1] + abs(mohrAttrib->minScale),
                      mohrAttrib->sigmas[2] + abs(mohrAttrib->minScale)};

   int
      min, mid, max; // indecies of min, mid, and max sigma values

   findMinMidMax( & min, & mid, & max, tmpSigmas, 3);

   // see than new value is not out of bounds
   if (newVal > mohrAttrib->maxScale)
      newVal = mohrAttrib->stressScaleMax;
   if (newVal < mohrAttrib->minScale)
      newVal = mohrAttrib->stressScaleMin;

   // right shift new value to match temp sigmas
   // newVal = newVal + abs(mohrAttrib->minScale);

   // figure the new values using the ratios
   tmpSigmas[mid] = newVal + abs(mohrAttrib->minScale);
   tmpSigmas[max] = ((1 / mohrAttrib->s2Tos1Ratio) * newVal) + abs(mohrAttrib->min
Scale);
   tmpSigmas[min] = (mohrAttrib->s3Tos1Ratio * ((1 / mohrAttrib->s2Tos1Ratio) * ne
wVal)) + abs(mohrAttrib->minScale);

   if ( (tmpSigmas[mid] >= tmpSigmas[max]) || (tmpSigmas[mid] <= tmpSigmas[min]) )
      return; // if bad values, then abort changes

   // correct sigmas are tmp sigmas left shifted
```

```
    mohrAttrib->sigmas[0] = tmpSigmas[0] - abs(mohrAttrib->minScale);
    mohrAttrib->sigmas[1] = tmpSigmas[1] - abs(mohrAttrib->minScale);
    mohrAttrib->sigmas[2] = tmpSigmas[2] - abs(mohrAttrib->minScale);

    // check the bounds on the new sigmas
    checkStressBounds();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  changeMinStress (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newVal -- value to store as new max sigma
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to newVal, mohrAttrib->s2Tos1Ratio, and
//                  mohrAttrib->s3Tos1Ratio
//
//  Purpose:  This function is used to handle changes in stress when the
//            stresses depend on each other.  It is called from
//            mohrScaleChange
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void changeMinStress(double newVal)
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   // tmpSigmas are sigmas right shifted to handle negatives as
   // lesser positives so that ratios work correctly
   double
      tmpSigmas[3] = {mohrAttrib->sigmas[0] + abs(mohrAttrib->minScale),
                      mohrAttrib->sigmas[1] + abs(mohrAttrib->minScale),
                      mohrAttrib->sigmas[2] + abs(mohrAttrib->minScale)};

   int
      min, mid, max; // indecies of min, mid, and max sigma values

   findMinMidMax( & min, & mid, & max, tmpSigmas, 3);

   // see than new value is not out of bounds
   if (newVal > mohrAttrib->maxScale)
      newVal = mohrAttrib->stressScaleMax;
   if (newVal < mohrAttrib->minScale)
      newVal = mohrAttrib->stressScaleMin;

   // right shift new value to match temp sigmas
   // newVal = newVal + abs(mohrAttrib->minScale);

   // figure the new values using the ratios
   tmpSigmas[min] = newVal + abs(mohrAttrib->minScale);
   tmpSigmas[max] = ((1 / mohrAttrib->s3Tos1Ratio) * newVal) + abs(mohrAttrib->min
Scale);
   tmpSigmas[mid] = (mohrAttrib->s2Tos1Ratio * ((1 / mohrAttrib->s3Tos1Ratio) * ne
wVal)) + abs(mohrAttrib->minScale);

   if (tmpSigmas[min] >= tmpSigmas[mid]) // if new values are bad
      return;  // then abort changes

   // correct sigmas are tmp sigmas left shifted
   mohrAttrib->sigmas[0] = tmpSigmas[0] - abs(mohrAttrib->minScale);
   mohrAttrib->sigmas[1] = tmpSigmas[1] - abs(mohrAttrib->minScale);
   mohrAttrib->sigmas[2] = tmpSigmas[2] - abs(mohrAttrib->minScale);

   // check the bounds on the new sigmas
   checkStressBounds();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```



```
//  Function:  changeStressBounds (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  none
//
//  State changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  according to bound violated, mohrAttrib->s2Tos1Ratio, and
//                  mohrAttrib->s3Tos1Ratio
//
//  Purpose:  This function is used to handle stresses that go beyond the
//            valid bounds when stresses depend on each other and are changed.
//            It is called from each of the changeM??Stress functions
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void checkStressBounds()
{
   MohrAttributeType // get the attributes of the Mohr graph
      * mohrAttrib = mohrObj.getAttributes();

   int
      boundFlag = -1;  // to tell which bound was crossed
                       //  -1 == no bound problem
                       //   0 == upper bound violated
                       //   1 == lower bound violated
   int
      i, // loop index
      min, mid, max; // indecies of min, mid, and max sigma values

   // get the indecies of min, mid, and max sigma values
   findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3);

   for(i = 0; i < 3; i++)
   {
      if (mohrAttrib->sigmas[i] > mohrAttrib->maxScale)
         boundFlag = 0;
      if (mohrAttrib->sigmas[i] < mohrAttrib->minScale)
         boundFlag = 1;
   }

   if (boundFlag == 0)
   {
      changeMaxStress(mohrAttrib->maxScale);
   }

   if (boundFlag == 1)
   {
      changeMinStress(mohrAttrib->minScale);
   }
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  findMinMidMax (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  int * min, int * mid, int * max, -- addresses to store info in
//              double values[] -- values to get min, mid, max indecies of
//              int noelts -- number of elements in values[]
//
//  State changes:  min, mid, and max are changed to reflect the indecies of
//                  minimum, middle, and maximum values in int values[]
//
//  Purpose:  This function is used to find the indecies of minimum, middle,
//            and maximum values in double values[]
//
// Last Modified:  9 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts)
{
```

Jul 10 1997 16:10 **10July97.log** Page 5

```
    int i;   // loop index
    *min = *mid = *max = 0;       // initialize variables

    for(i = 0; i < noelts; i++) // find the max and min
    {
       if (values[i] > values[*max])
          *max = i; // new max has been found
       if (values[i] < values[*min])
          *min = i; // new min has been found
    }

    for(i = 0; i < noelts; i++)  // find mid
    {
       if( (i != *min) && (i != *max) )
          *mid = i; // mid has been found
    }

    if(min == max) // make sure every index is unique
    {
       for(i = 0; i < noelts; i++)  // find mid
       {
          if( (i != *mid) && (i != *max) )
             *min = i; // mid has been found
       }
    }
}
```


Jul 11 1997 12:46 **11July97.log** Page 1

mohrOptionCB.hh

Added a new function to find indecies of extremes in arrays:

```
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts);
```

mohrOptionCB.c++

Added a new function to find indecies of extremes in arrays:

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  findMinMidMax (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  int * min, int * mid, int * max, -- addresses to store info in
//              double values[] -- values to get min, mid, max indecies of
//              int noelts -- number of elements in values[]
//
//  State changes:  min, mid, and max are changed to reflect the indecies of
//                  minimum, middle, and maximum values in int values[]
//
//  Purpose:  This function is used to find the indecies of minimum, middle,
//            and maximum values in double values[]
//
// Last Modified:  11 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts)
{
   int i;   // loop index
   *min = *mid = *max = 0;       // initialize variables

   for(i = 0; i < noelts; i++) // find the max and min
   {
      if (values[i] > values[*max])
         *max = i; // new max has been found
      if (values[i] < values[*min])
         *min = i; // new min has been found
   }

   for(i = 0; i < noelts; i++)  // find mid
   {
      if( (i != *min) && (i != *max) )
         *mid = i; // mid has been found
   }
}
```

Tinkered with mohrScaleChange to correct error with dependent stresses

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              Ask Robert Boenau what the hell these are.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigma
//                  that was changed in the Mohr graph options window
//                  by the user.  Scales in the Options window are
//                  changed.
//
//  Purpose:  This program updates the sigmas according to the settings
//            of the scales in the Mohr graph options window devoted to
//            the sigmas.
//
// Last Modified:  16 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
   XmScaleCallbackStruct
```

```
    * cbs = (XmScaleCallbackStruct *) callData;

//  The variable below holds the address to mohrObj's
//  attributes private member variable.
MohrAttributeType
   * mohrObjAttrib = mohrObj.getAttributes();

//  The variable below holds the member variables of the
//  class responsible for the Mohr Graph's Options window
MohrOptionType *optionsAttrib = mohrOptionObj.getAttributes();

//  The variable below, whichSig, is the index of the sigma that
//  was changed in the Mohr's options window.  Also, newValue holds
//  the value that the user set to the sigma indicated by whichSig,
//  and delta holds the change in the sigma value.
int
   whichSig = (int) clientData,
   max, mid, min,  // indecies of largest, middle, & smallest sigma values
   max2, mid2, min2; // post-change indecies for comparison

double
   oldSigmas[3];  // holder for sigma values previous to change
   newValue = (double) cbs->value / 1000,
   delta = newValue - mohrObjAttrib->sigmas[whichSig];

if(mohrObjAttrib->stressMode == 0)
[  //stresses should depend on each other
   //store old sigma values
   oldSigmas[0] = mohrObjAttrib->sigmas[0];
   oldSigmas[1] = mohrObjAttrib->sigmas[1];
   oldSigmas[2] = mohrObjAttrib->sigmas[2];

   // Search for the largest and smallest sigmas
   findMinMidMax( & min, & mid, & max, oldSigmas, 3 );

   if ( whichSig == max )
      changeMaxStress(1, max, mid, min, delta, 1);

   else if ( whichSig == min )
      changeMinStress(1, max, mid, min, delta, 1);

   else  // middle value changed
      changeMidStress(1, max, mid, min, delta, 1);

   findMinMidMax( & min2, & mid2, & max2, mohrObjAttrib->sigmas, 3 );

   if ( (max != max2) || (min != min2) || (max != max2) )
   {  // if the sigmas got out of order
      mohrObjAttrib->sigmas[0] = oldSigmas[0];  // restore old sigmas
      mohrObjAttrib->sigmas[1] = oldSigmas[1];
      mohrObjAttrib->sigmas[2] = oldSigmas[2];
      return;  // abort changes
   }

   else
   {
      XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max] * 1
000);
      XmScaleSetValue(optionsAttrib->scale[mid], mohrObjAttrib->sigmas[mid] * 1
000);
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min] * 1
000);
   }
}// end if(mohrObjAttrib->stressMode == 0)

else  // scales should be inDependent
{
   mohrObjAttrib->sigmas[whichSig] = newValue;
}  // end else

updateEffective();
```



```
   mohrObj.failure();
   mohrObj.display();
} // end of mohrScaleChange

An alternative method was tried using the ratios of the dependent stresses:

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChanges3Tos1RatioField (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  mohrObj.attributes->s3Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  2 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChanges3Tos1RatioField(Widget, XtPointer, XtPointer)
{
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();
   MohrAttributeType // get Mohr graph stats
      * mohrObjAttrib = mohrObj.getAttributes();

   char
      strHolder[100], // holder for transfer of numbers to ASCII
      * val = XmTextFieldGetString(moattr->s3Tos1RatioText); // get new value

   double
      delta, ratio,
      tmpValue;  // holder for new value in numeric form

   int
      max, mid, min;

   if (strlen(val) == 0) // is field is completely blank
   {  // tell user of mistake
      infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
      sprintf(strHolder,"%5lf", mohrObjAttrib->s3Tos1Ratio);    // reset field
      XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
      return;
   }

   tmpValue = atof(val);  // convert new value to numeric form

   if ( ((int) (tmpValue * 100) > mohrObjAttrib->stressScaleMax) || ((int) (tmpVal
ue * 100) < mohrObjAttrib->stressScaleMin) )
   {  // check that new value is within bounds
         infoWidget(moattr->mohrDialogForm,"Ratio out of bounds");  // dialog box
error message
         sprintf(strHolder,"%5lf", mohrObjAttrib->s3Tos1Ratio);  // reset field
         XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
         return;
   }

   mohrObjAttrib->s3Tos1Ratio = tmpValue; // update variable
   // update scale
   XmScaleSetValue(moattr->s3Tos1RatioScale, (int) (mohrObjAttrib->s3Tos1Ratio * 1
00));

   findMinMidMax( & min, & mid, & max, mohrObjAttrib->sigmas, 3 );

   delta = mohrObjAttrib->sigmas[min];

   ratio = mohrObjAttrib->s3Tos1Ratio * mohrObjAttrib->sigmas[max];
   ratio = (ratio / mohrObjAttrib->sigmas[min]) - 1;
```

```
    changeMinStress(0, max, mid, min, delta, ratio);

    XmScaleSetValue(moattr->scale[min], mohrObjAttrib->sigmas[min] * 1000);

    updateEffective();
    mohrObj.failure();
    mohrObj.display();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChanges2Tos1RatioField (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  mohrObj.attributes->s2Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  2 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChanges2Tos1RatioField(Widget, XtPointer, XtPointer)
{
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();
   MohrAttributeType // get Mohr graph stats
      * mohrObjAttrib = mohrObj.getAttributes();

   char
      strHolder[100], // holder for transfer of numbers to ASCII
      * val = XmTextFieldGetString(moattr->s2Tos1RatioText); // get new value

   double
      delta, ratio,
      tmpValue;  // holder for new value in numeric form

   int
      max, mid, min;

   if (strlen(val) == 0) // is field is completely blank
   {  // tell user of mistake
      infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
      sprintf(strHolder,"%5lf", mohrObjAttrib->s2Tos1Ratio);    // reset field
      XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
      return;
   }

   tmpValue = atof(val);  // convert new value to numeric form

   if ( ((int) (tmpValue * 100) > mohrObjAttrib->stressScaleMax) || ((int) (tmpVal
ue * 100) < mohrObjAttrib->stressScaleMin) )
   {  // check that new value is within bounds
         infoWidget(moattr->mohrDialogForm,"Ratio out of bounds");  // dialog box
error message
         sprintf(strHolder,"%5lf", mohrObjAttrib->s2Tos1Ratio);  // reset field
         XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
         return;
   }

   mohrObjAttrib->s2Tos1Ratio = tmpValue; // update variable
   // update scale
   XmScaleSetValue(moattr->s2Tos1RatioScale, (int) (mohrObjAttrib->s3Tos1Ratio * 1
00));

   delta = mohrObjAttrib->sigmas[mid];
```



```
    ratio = mohrObjAttrib->s2Tos1Ratio * mohrObjAttrib->sigmas[max];
    ratio = (ratio / mohrObjAttrib->sigmas[mid]) - 1;

    changeMidStress(0, max, mid, min, delta, ratio);

    XmScaleSetValue(moattr->scale[mid], mohrObjAttrib->sigmas[mid] * 1000);

    updateEffective();
    mohrObj.failure();
    mohrObj.display();
}
```

mohrClass.c++

Changed the initialization for dependent stress ratios:

```
attributes.s3Tos1Ratio = 0.10; // ratios initially 25%
attributes.s2Tos1Ratio = 0.25; // ratios initially 25%
```

mohrOptionCB.hh

Removed previous stress changing functions and added one and a function to find min, mid, and max sigma indecies:

```
#ifndef __MOHROPTIONCB_HH__
#define __MOHROPTIONCB_HH__

void mohrChangeLineWidth(Widget, XtPointer, XtPointer);
void recalcSigmas(int min, int mid, int max, double newMax);
void mohrScaleChange(Widget, XtPointer, XtPointer);
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts);
```

Completely reworked the way dependent stress works. Dependent stress now operates off strait ratios instead of Mendal's method.

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  recalcSigmas (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newMax -- The previous maximum sigma value takes
//              this as its new value.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  with respect to newMax and the dependent stress
//                  ratios.
//
//  Purpose:  This program updates the sigmas according to newMax and
//            the dependent stress ratios.
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void recalcSigmas(int min, int mid, int max, double newMax)
{
   MohrAttributeType // mohr graph's attributes
      * mattr = mohrObj.getAttributes();

   if (newMax > mattr->maxScale) // if the new value is larger than possible
      newMax = mattr->maxScale;  // scale it down to maximum

   if (newMax <= 0)  // ratios only work with positive stresses
      newMax = 1;    // also, min, mid, and max stresses should not trade places

   mattr->sigmas[max] = newMax;  // now, set the new values according to ratios
   mattr->sigmas[mid] = newMax * mattr->s2Tos1Ratio;
   mattr->sigmas[min] = newMax * mattr->s3Tos1Ratio;
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              Ask Robert Boenau what the hell these are.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigma
//                  that was changed in the Mohr graph options window
//                  by the user.  It calls recalcSigmas to accomplish
//                  this.  Scales in the Options window are changed.
//
//  Purpose:  This program updates the sigmas according to the settings
//            of the scales in the Mohr graph options window devoted to
```



```
//            the sigmas.
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mohrObjAttrib = mohrObj.getAttributes();

   //  The variable below holds the member variables of the
   //  class responsible for the Mohr Graph's Options window
   MohrOptionType *optionsAttrib = mohrOptionObj.getAttributes();

   //  The variable below, whichSig, is the index of the sigma that
   //  was changed in the Mohr's options window.  Also, newValue holds
   //  the value that the user set to the sigma indicated by whichSig,
   //  and delta holds the change in the sigma value.
   int
      whichSig = (int) clientData,
      max, mid, min;  // indecies of largest, middle, & smallest sigma values

   double
      newValue = (double) cbs->value / 1000;

   if(mohrObjAttrib->stressMode == 0)
   {  //stresses should depend on each other
      // Search for the largest and smallest sigmas
      findMinMidMax( & min, & mid, & max, mohrObjAttrib->sigmas, 3 );

      if ( whichSig == max )
         recalcSigmas(min, mid, max, newValue);

      else if ( whichSig == mid ) // user changed middle value
         recalcSigmas(min, mid, max, newValue * (1 / mohrObjAttrib->s2Tos1Ratio));

      else  // minimum value changed
         recalcSigmas(min, mid, max, newValue * (1 / mohrObjAttrib->s3Tos1Ratio));

      XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max] * 1000
);
      XmScaleSetValue(optionsAttrib->scale[mid], mohrObjAttrib->sigmas[mid] * 1000
);
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min] * 1000
);
   }// end if(mohrObjAttrib->stressMode == 0)

   else  // scales should be inDependent
   {
      mohrObjAttrib->sigmas[whichSig] = newValue;
   }  // end else

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
} // end of mohrScaleChange

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  findMinMidMax (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  int * min, int * mid, int * max, -- addresses to store info in
//              double values[] -- values to get min, mid, max indecies of
//              int noelts -- number of elements in values[]
//
//  State changes:  min, mid, and max are changed to reflect the indecies of
```

```
//                  minimum, middle, and maximum values in int values[]
//
//  Purpose:  This function is used to find the indecies of minimum, middle,
//            and maximum values in double values[]
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void findMinMidMax(int * min, int * mid, int * max, double values[], int noelts)
{
   int i;   // loop index
   *min = *mid = *max = 0;       // initialize variables

   for(i = 0; i < noelts; i++) // find the max and min
   {
      if (values[i] >= values[(*max)])
         *max = i; // new max has been found
      if (values[i] <= values[(*min)])
         *min = i; // new min has been found
   }

   for(i = 0; i < noelts; i++)  // find mid
   {
      if( (i != *min) && (i != *max) )
         *mid = i; // mid has been found
   }
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChanges3Tos1RatioField (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  mohrObj.attributes->s3Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChanges3Tos1RatioField(Widget, XtPointer, XtPointer)
{
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();
   MohrAttributeType // get Mohr graph stats
      * mattr = mohrObj.getAttributes();

   char
      strHolder[100], // holder for transfer of numbers to ASCII
      * val = XmTextFieldGetString(moattr->s3Tos1RatioText); // get new value

   double
      tmpValue;  // holder for new value in numeric form
   int
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

   if (strlen(val) == 0) // is field is completely blank
   {  // tell user of mistake
      infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
      sprintf(strHolder,"%5lf", mattr->s3Tos1Ratio);     // reset field
      XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
      return;
   }

   tmpValue = atof(val);  // convert new value to numeric form
```



```
   if ( ((int) (tmpValue * 100) > mattr->stressScaleMax) || ((int) (tmpValue * 100
) < mattr->stressScaleMin) )
   {  // check that new value is within bounds
      infoWidget(moattr->mohrDialogForm,"Ratio out of bounds");  // dialog box
error message
      sprintf(strHolder,"%5lf", mattr->s3Tos1Ratio);  // reset field
      XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
      return;
   }

   if ( tmpValue > mattr->s2Tos1Ratio )
   {  // check that new value is within bounds
      infoWidget(moattr->mohrDialogForm,"Ratio violates sigma orders.");  // di
alog box error message
      sprintf(strHolder,"%5lf", mattr->s3Tos1Ratio);  // reset field
      XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);
      return;
   }

   mattr->s3Tos1Ratio = tmpValue; // update variable
   // update scale
   XmScaleSetValue(moattr->s3Tos1RatioScale, (int) (mattr->s3Tos1Ratio * 100));

   recalcSigmas(min, mid, max, mattr->sigmas[max]);
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 1000);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 1000);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 1000);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrChanges2Tos1RatioField (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  mohrObj.attributes->s2Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrChanges2Tos1RatioField(Widget, XtPointer, XtPointer)
{
   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();
   MohrAttributeType // get Mohr graph stats
      * mattr = mohrObj.getAttributes();

   char
      strHolder[100], // holder for transfer of numbers to ASCII
      * val = XmTextFieldGetString(moattr->s2Tos1RatioText); // get new value

   double
      tmpValue;  // holder for new value in numeric form
   int
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

   if (strlen(val) == 0) // is field is completely blank
   {  // tell user of mistake
      infoWidget(moattr->mohrDialogForm,"Not allowed to have empty value");
      sprintf(strHolder,"%5lf", mattr->s2Tos1Ratio);     // reset field
```

```
      XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
      return;
   }

   tmpValue = atof(val);  // convert new value to numeric form

   if ( ((int) (tmpValue * 100) > mattr->stressScaleMax) || ((int) (tmpValue * 100
) < mattr->stressScaleMin) )
   {  // check that new value is within bounds
      infoWidget(moattr->mohrDialogForm,"Ratio out of bounds");  // dialog box
error message
      sprintf(strHolder,"%5lf", mattr->s2Tos1Ratio);  // reset field
      XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
      return;
   }

   if ( tmpValue < mattr->s3Tos1Ratio )
   {  // check that new value is within bounds
      infoWidget(moattr->mohrDialogForm,"Ratio violates sigma orders.");  // di
alog box error message
      sprintf(strHolder,"%5lf", mattr->s2Tos1Ratio);  // reset field
      XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);
      return;
   }

   mattr->s2Tos1Ratio = tmpValue; // update variable
   // update scale
   XmScaleSetValue(moattr->s2Tos1RatioScale, (int) (mattr->s2Tos1Ratio * 100));

   recalcSigmas(min, mid, max, mattr->sigmas[max]);
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 1000);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 1000);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 1000);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrs3Tos1RatioScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.attributes->s3Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrs3Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mattr = mohrObj.getAttributes();

   char
      strHolder[100]; // holder for transfer of numbers to ASCII
   int
```



```
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

   // update the ratio
   mattr->s3Tos1Ratio =  (double) (cbs->value / 100.0);

   if (mattr->s3Tos1Ratio > mattr->s2Tos1Ratio)
   {
      mattr->s3Tos1Ratio = mattr->s2Tos1Ratio - 0.001;
      XmScaleSetValue(moattr->s3Tos1RatioScale, (int) (mattr->s3Tos1Ratio * 100));
   }

   sprintf(strHolder,"%5lf", mattr->s3Tos1Ratio);  // reset field
   XmTextFieldSetString(moattr->s3Tos1RatioText, strHolder);

   recalcSigmas(min, mid, max, mattr->sigmas[max]);
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 1000);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 1000);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 1000);

   updateEffective();
   mohrObj.failure();
   mohrObj.display();
}  // end of mohrs3Tos1RatioScaleChange

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrs2Tos1RatioScaleChange (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer callData
//
//  State Changes:  mohrObj.attributes->s2Tos1Ratio, updates the degree
//                  to which the stress are dependent when in dependent
//                  stress mode
//
//  Purpose:  This program updates the allows the user to change Poisson's
//            ratio in the calculation of dependency for dependent stresses
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrs2Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
{
   XmScaleCallbackStruct
      * cbs = (XmScaleCallbackStruct *) callData;

   MohrOptionType // get option window's stats
      * moattr = mohrOptionObj.getAttributes();

   //  The variable below holds the address to mohrObj's
   //  attributes private member variable.
   MohrAttributeType
      * mattr = mohrObj.getAttributes();

   char
      strHolder[100]; // holder for transfer of numbers to ASCII
   int
      min, mid, max;

   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

   // update the ratio
   mattr->s2Tos1Ratio =  (double) (cbs->value / 100.0);

   if (mattr->s2Tos1Ratio < mattr->s3Tos1Ratio)
   {
      mattr->s2Tos1Ratio = mattr->s3Tos1Ratio + 0.001;
      XmScaleSetValue(moattr->s2Tos1RatioScale, (int) (mattr->s2Tos1Ratio * 100));
   }
```

```
    sprintf(strHolder,"%5lf", mattr->s2Tos1Ratio);  // reset field
    XmTextFieldSetString(moattr->s2Tos1RatioText, strHolder);

    recalcSigmas(min, mid, max, mattr->sigmas[max]);
    XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 1000);
    XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 1000);
    XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 1000);

    updateEffective();
    mohrObj.failure();
    mohrObj.display();
}  // end of mohrs2Tos1RatioScaleChange

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrToggleStressMode (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              clientData is the data that represents the action the
//              user took.  Ask Robert Boenau what the hell the rest are.
//
//  State Changes:  mohrObj.attributes->stressMode, changes the stress
//                  mode between dependent stresses and independent
//                  stresses. When stress is set to independent, the stress
//                  dependency ratio scale is greyed out;  when it is set to
//                  dependent, the stress dependency ratio is activated.
//                  When stresses are set to dependent, sigmas are recalculated
//                  to take ratios into account.
//
//  Purpose:  This function switches between the two stress modes
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrToggleStressMode(Widget, XtPointer clientData, XtPointer)
{
   // get the attributes of the current Mohr graph for changing
   MohrAttributeType *mohrAttrib = mohrObj.getAttributes();

   // get the attributes of the option window (widgets and such)
   MohrOptionType *optionAttrib = mohrOptionObj.getAttributes();

   // set the stress mode to the appropriate value (0 or 1)
   // stressMode will be 0 for dependent stresses and 1 for independent
   mohrAttrib->stressMode = (int)clientData;

   int
      min, mid, max;

   if (mohrAttrib->stressMode == 1)  // if stress is independent, grey out the rat
io scale
   {
      XtSetSensitive(optionAttrib->s3Tos1RatioText, False);
      XtSetSensitive(optionAttrib->s2Tos1RatioText, False);
      XtSetSensitive(optionAttrib->s3Tos1RatioScale, False);
      XtSetSensitive(optionAttrib->s2Tos1RatioScale, False);
   }
   else  // if stress is dependent, activate the ratio scale
   {
      findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3 );
      recalcSigmas(min, mid, max, mohrAttrib->sigmas[max]);

      XmScaleSetValue(optionAttrib->scale[0], mohrAttrib->sigmas[0] * 1000);
      XmScaleSetValue(optionAttrib->scale[1], mohrAttrib->sigmas[1] * 1000);
      XmScaleSetValue(optionAttrib->scale[2], mohrAttrib->sigmas[2] * 1000);

      XtSetSensitive(optionAttrib->s3Tos1RatioText, True);
      XtSetSensitive(optionAttrib->s2Tos1RatioText, True);
      XtSetSensitive(optionAttrib->s3Tos1RatioScale, True);
      XtSetSensitive(optionAttrib->s2Tos1RatioScale, True);
```



```
      updateEffective();
      mohrObj.failure();
      mohrObj.display();
   }
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  updateEffective (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  void
//
//  State Changes:  mohrObj.attributes->effStress are changed and this change
//                  is displayed on screen in the text fields
//
//  Purpose:  This function takes care of calculating effective stress
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void updateEffective()
{
   MohrOptionType  // option window attributes
      *moattr = mohrOptionObj.getAttributes();
   MohrAttributeType  // mohr graph attributes
      *mattr = mohrObj.getAttributes();

   char strs[3][100]; // holder for transfer of numbers to ASCII

   int
      i,    // loop index
      min, mid, max; // hold indeces of smallest, middle, and largest sigmas

   if (mattr->stressMode == 0) // stresses are dependent
   {
      findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

      // we are now ready to set the effective stresses
      mattr->effStress[max] = (double)mattr->sigmas[max] - mohrObj.fluidPressure;
      mattr->effStress[mid] = (double)mattr->sigmas[mid] - (mattr->s2Tos1Ratio * m
ohrObj.fluidPressure);
      mattr->effStress[min] = (double)mattr->sigmas[min] - (mattr->s3Tos1Ratio * m
ohrObj.fluidPressure);
   } // end if (mattr->stressMode == 0)

   else // stresses are independent so set them strait
   {
      mattr->effStress[0] = (double)mattr->sigmas[0] - mohrObj.fluidPressure;
      mattr->effStress[1] = (double)mattr->sigmas[1] - mohrObj.fluidPressure;
      mattr->effStress[2] = (double)mattr->sigmas[2] - mohrObj.fluidPressure;
   } // end else

   for (i = 0; i < 3; i++)  // set the scales on the screen
   {
      // check bounds
      if (mattr->effStress[i] > mattr->maxScale)
      {
         if(moattr->isInfoWinOpen == FALSE)
         {  // if there is no current error win open
            moattr->isInfoWinOpen = TRUE; // open the window and tell the world ab
out it
            infoWidget(moattr->mohrDialogForm, "Warning:  Actual Effective Stress
Value out of bounds.\nSetting value to an artifical bound value.", closeBoundsError
Win);
         }
         mattr->effStress[i] = mattr->maxScale;
      }

      if (mattr->effStress[i] < mattr->minScale)
      {
         if(moattr->isInfoWinOpen == FALSE)
```

```
         {  // if there is no current error win open
            moattr->isInfoWinOpen = TRUE; // open the window and tell the world ab
out it
            infoWidget(moattr->mohrDialogForm, "Warning:  Actual Effective Stress
Value out of bounds.\nSetting value to an artifical bound value.", closeBoundsError
Win);
         }
         mattr->effStress[i] = mattr->minScale;
      }
      sprintf(strs[i],"%5lf",mattr->effStress[i]);
      XmTextFieldSetString(moattr->effText[i], strs[i]);
   } // end for int i

   updateEffRatios();  // update the effective stress ratio text boxes
} // end of updateEffective

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  updateEffRatios (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  void
//
//  State Changes:  the effective ratio text fields are changed and that
//                  is all
//
//  Purpose:  This function takes care of calculating effective stress
//            ratios
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void updateEffRatios()
{
   MohrOptionType  // option window attributes
      *moattr = mohrOptionObj.getAttributes();
   MohrAttributeType  // mohr graph attributes
      *mattr = mohrObj.getAttributes();
   int
      min, mid, max; // hold indeces of smallest, middle, and largest sigmas
   double holder; // holder for storing ratios
   char str[100]; // holder for string that goes in text boxes

   findMinMidMax( & min, & mid, & max, mattr->sigmas, 3 );

   if(mattr->effStress[max] != 0.0)   // can't divide by 0.0
   {
      holder = (double) (mattr->effStress[min] / mattr->effStress[max]);
      sprintf(str, "%5lf", holder);  // make the number a string
   }
   else  // division by zero is not defined in our system of mathematics
      sprintf(str, "%s", "Undefined\0");
   // update the text box
   XmTextFieldSetString(moattr->effs3Tos1RatioText, str);

   if(mattr->effStress[max] != 0)   // can't divide by 0.0
   {
      holder = (double) (mattr->effStress[mid] / mattr->effStress[max]);
      sprintf(str, "%5lf", holder); // make the number a string
   }
   else  // division by zero is not defined in our system of mathematics
      sprintf(str, "%s", "Undefined\0");
   // update the text box
   XmTextFieldSetString(moattr->effs2Tos1RatioText, str);
}
```


mohrOptionCB.c++

Added ending comments to all functions. And fixed most of the code's unnecessarily long lines.

Changed recalcSigs to deal with the fact that fluid pressure causes displayed stresses to interchange under the pervious bounds check and correction:

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  recalcSigmas (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  double newMax -- The previous maximum sigma value takes
//                this as its new value.
//
//  State Changes:  mohrObj.attributes->sigmas[?], updates the sigmas
//                  with respect to newMax and the dependent stress
//                  ratios.
//
//  Purpose:  This program updates the sigmas according to newMax and
//            the dependent stress ratios.
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void recalcSigmas(int min, int mid, int max, double newMax)
{
   MohrAttributeType // mohr graph's attributes
      * mattr = mohrObj.getAttributes();

   if (newMax > mattr->maxScale) // if the new value is larger than possible
      newMax = mattr->maxScale;  // scale it down to maximum

   // ratios only work with positive effective stresses
   if (newMax - mohrObj.fluidPressure <= 0) // effstress = sigma - fluidP.
      newMax = 0.01 + mohrObj.fluidPressure;
   // also, min, mid, and max stresses shouldn't trade places

   mattr->sigmas[max] = newMax; // now, set the new values according to ratios
   mattr->sigmas[mid] = newMax * mattr->s2Tos1Ratio;
   mattr->sigmas[min] = newMax * mattr->s3Tos1Ratio;
} // end function recalcSigmas
```

Changed the bounds check in the fluid text call back function to avoid bounds problems on the sigmas:

```
 if (tmpValue > mohrObjAttrib->maxScale - 1)
```

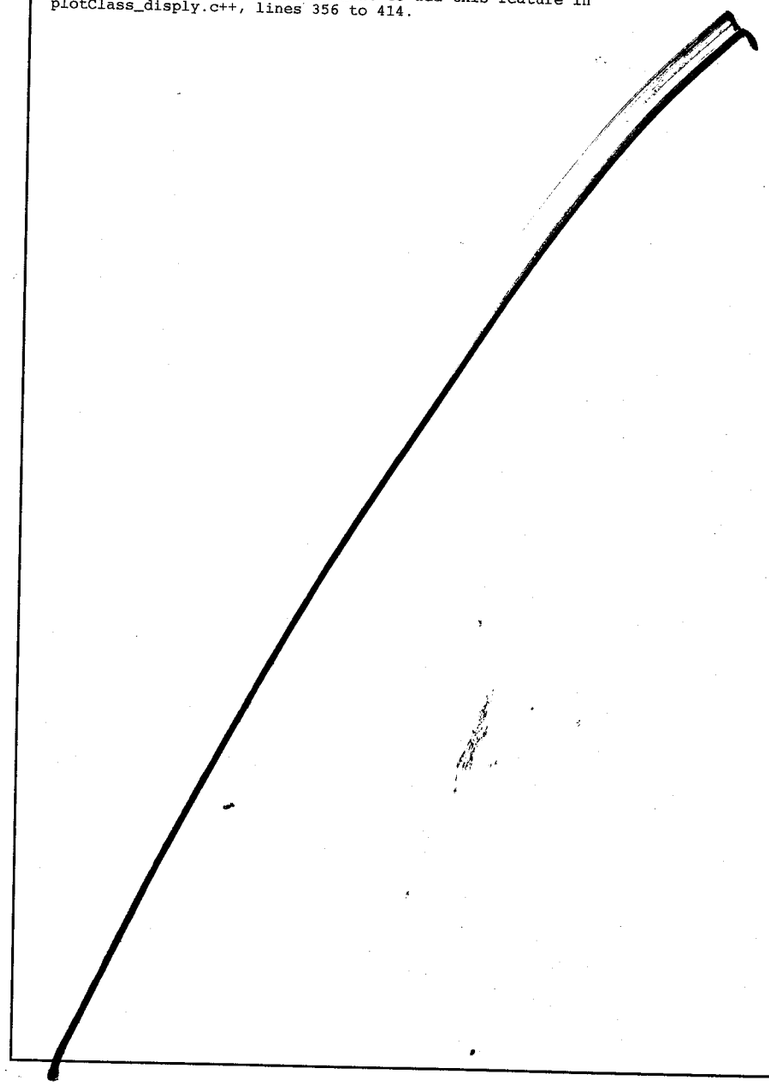
mohrOptionClass.c++

Changed fluidScale to avoid running out of bounds on the sigmas:

```
   attributes.fluidScale = XtVaCreateManagedWidget("fluidScale",
                              xmScaleWidgetClass, col3,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, mattr->maxScale - 1,
                              XmNminimum, 0,
                              XmNvalue, (int) mohrObj.fluidPressure,
                              XmNscaleMultiple, 1,
                              XmNshowValue, False,
                              XmNbackground, WIDGET_COLOR,  NULL);
```

Jul 18 1997 08:25 **16July97.log** Page 1

Discussed the possibility of adding a color scheme to 3dstress for plotting the ratio of fluid pressure over normal stress. Found the area of code in which most of the work must be done to add this feature in plotClass_disply.c++, lines 356 to 414.



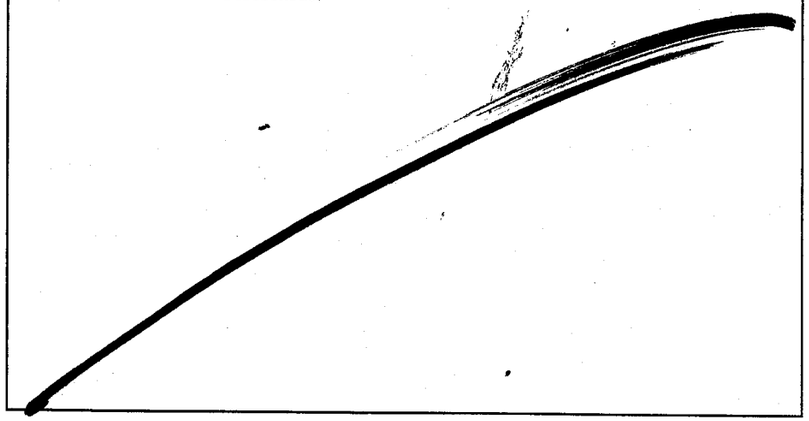
Jul 18 1997 08:30 **17July97.log** Page 1

plogClass_display.c++

Added another display case called fluid which displays a plot of the ratio of fluid pressure to normal stress at each dot.

```
            if (attributes.displayMode == StressRatio) {
               if (sigma1 == sigma3) {
                     stressRatio = 0.0;
                     dotData[i] = stressRatio;
                     dotDataMax = 1.0;
                     dotDataMin = 0.0;
               } else {
                  if (normalStress == 0.0) {
                     stressRatio = shearStress * 100000.0;
                     // cout << "***** NORMAL STRESS = 0.0 ******"
                     //      << endl << flush;
                     // cout << "SR " << stressRatio << endl << flush;
                  } else {
                     stressRatio = shearStress/normalStress;
                  }
                  dotData[i] = stressRatio;
                  if (stressRatio > dotDataMax) dotDataMax = stressRatio;
                  if (stressRatio < dotDataMin) dotDataMin = stressRatio;
               }
            } else if (attributes.displayMode == Fluid){
               if (sigma1 == sigma3) {
                  dotData[i] = fluidCoef = 0.0;
                  dotDataMax = 1.0;
                  dotDataMin = fluidCoef;
               }
               else {
                  if (normalStress == 0.0)
                     fluidCoef = 0.0;
                  else
                     fluidCoef = attributes.fluidPressure/ normalStress;

                  dotData[i] = fluidCoef;

                  if (fluidCoef > dotDataMax)
                     dotDataMax = fluidCoef;
                  if (fluidCoef < dotDataMin)
                     dotDataMin = fluidCoef;
               }
            } else {
               if (sigma1 == sigma3) {
                  dilationCoef = 0.0;
                  dotData[i] = dilationCoef;
                  dotDataMax = 1.0;
                  dotDataMin = dilationCoef;
                  // cout << "***** DILATION COEFF / 0.0 ******"
                  //      << endl << flush;
                  // cout << "NS " << normalStress << endl << flush;
               } else {
                  dilationCoef = (sigma1-normalStress)/(sigma1-sigma3);
                  dotData[i] = dilationCoef;
                  if (dilationCoef > dotDataMax) dotDataMax = dilationCoef;
                  if (dilationCoef < dotDataMin) dotDataMin = dilationCoef;
               }
            }
```

```
                    mohrToggleStressMode, (XtPointer)0);
   XtAddCallback (attributes.stressIndependantRadio, XmNvalueChangedCallback,
                    mohrToggleStressMode, (XtPointer)1);
// End of Call backs to change the stress mode between dependant and independant


mohrOptionCB.hh:  (line 33) added function prototype for stress dependency radio b
ox

   void mohrToggleStressMode(Widget, XtPointer clientData, XtPointer);

mohrOptionCB.c++:  added callback function for the stress dependency radio box

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrToggleStressMode (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              clientData is the data that represents the action the
//              user took.  Ask Robert Boenau what the hell the rest are.
//
//  State Changes:  mohrObj.attributes->stressMode, changes the stress
//                  mode between dependant stresses and independant
//                  stresses
//
//  Purpose:  This function switches between the two stress modes
//
// Last Modified:  17 June 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrToggleStressMode(Widget, XtPointer clientData, XtPointer)
{
   // get the attributes of the current Mohr graph for changing
   MohrAttributeType *mohrAttrib = mohrObj.getAttributes();

   // set the stress mode to the appropriate value (0 or 1)
   // stressMode will be 0 for Dependant stresses and 1 for independant
   mohrAttrib->stressMode = (int)clientData;
}

mohrClass.hh:  added variable for the controll of the new stress dependency radio
box

     int             stressMode;
```




```
optionClass.hh

   Widget            slipButton,
                     dilButton,
                     fluidButton,
                     fluidPress,
                     tensileStrScale,

optionClass.c++

      attributes.tensileStrScale = XtVaCreateManagedWidget(
                                          "Tensile Str",
                                          xmScaleWidgetClass,
                                          col1,
                                          XtVaTypedArg,
                                          XmNtitleString,
                                          XmRString, "Tensile Strength", 17,
                                          XmNorientation, XmHORIZONTAL,
                                          XmNmaximum, 20,
                                          XmNminimum, 0,
                                          XmNscaleMultiple, 1,
                                          XmNvalue, 10,
                                          XmNscaleHeight, 20,
                                          XmNscaleWidth, 150,
                                          XmNshowValue, True,
                                          XmNbackground,"WIDGET_COLOR,
                                          NULL);

      XtSetSensitive(attributes.tensileStrScale, False);

      XtAddCallback(attributes.tensileStrScale, XmNvalueChangedCallback,
                    tensileStrCB, NULL);
      XtAddCallback(attributes.tensileStrScale, XmNdragCallback, tensileStrCB,
                    NULL);

optionCallbacks.hh

void tensileStrCB(Widget, XtPointer, XtPointer);

optionCallbacks.c++

void changeOptionCompute(Widget,
                         XtPointer client_data,
                         XtPointer) {

   PlotAttrType            plotAttr;
   OptionAttributeType     *oattr = optionObj.getAttributes();

   plotAttr = plotObj.getAttributes();
   switch ((int)client_data) {
      case 1:
         plotAttr.displayMode = StressRatio;
         plotObj.setAttributes(plotAttr);
         sceneObj.update();
         XtSetSensitive(oattr->fluidPress, False);
         XtSetSensitive(oattr->tensileStrScale, False);
         break;
      case 2:
         plotAttr.displayMode = Dilation;
         plotObj.setAttributes(plotAttr);
         sceneObj.update();
         XtSetSensitive(oattr->fluidPress, False);
         XtSetSensitive(oattr->tensileStrScale, False);
         break;
      case 3:
         plotObj.toggleVerbose();
         XtSetSensitive(oattr->fluidPress, False);
         XtSetSensitive(oattr->tensileStrScale, False);
         break;
      case 4:
         XtSetSensitive(oattr->fluidPress, True);
```

```
            XtSetSensitive(oattr->tensileStrScale, True);
            plotAttr.displayMode = Fluid;
            plotObj.setAttributes(plotAttr);
            sceneObj.update();
            break;
         default:
            break;
      }
   } // endof changeOptionCompute

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  tensileStrCB (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget,
//              XtPointer callData -- new scale setting
//
//  State Changes:  plotObj.attributes->tensileStrength[?], updates the
//                  tensile strength that was changed in the options
//                  window by the user
//
//  Purpose:  This program updates the tensile strength according to the
//            setting of the scales in the Mohr graph options window
//            devoted to the sigmas.
//
// Last Modified:  18 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void tensileStrCB(Widget, XtPointer, XtPointer callData)
{
  XmScaleCallbackStruct // put new value in package
    * cbs = (XmScaleCallbackStruct *)callData;

  PlotAttrType plotAttr; // holder for plotObj attributes

  plotAttr = plotObj.getAttributes();  // get plotObj attributes
  plotAttr.tensileStrength = (float)(cbs->value); // set new value
  plotObj.setAttributes(plotAttr); // save changes
  sceneObj.update(); // update screen
} // endof tensileStrCB
```

plotClass.hh

```
    float             tensileStrength;
```

plotClass_display.c++

```
         drawText(attributes.textred,
                  attributes.textgrn,
                  attributes.textblu,
                   70, 125,
                  "FluidP/(NStress+TensileStr)");

            if (attributes.displayMode == StressRatio) {
               if (sigma1 == sigma3) {
                  stressRatio = 0.0;
                  dotData[i] = stressRatio;
                  dotDataMax = 1.0;
                  dotDataMin = 0.0;
               } else {
                  if (normalStress == 0.0) {
                  stressRatio = shearStress * 100000.0;
                  // cout << "***** NORMAL STRESS = 0.0 ******"
                  //      << endl << flush;
                  // cout << "SR " << stressRatio << endl << flush;
                  } else {
                    stressRatio = shearStress/normalStress;
                  }
                  dotData[i] = stressRatio;
                  if (stressRatio > dotDataMax) dotDataMax = stressRatio;
                  if (stressRatio < dotDataMin) dotDataMin = stressRatio;
```



```
            }
         } else if (attributes.displayMode == Fluid){
            if (sigma1 == sigma3) {
               dotData[i] = fluidCoef = 0.0;
               dotDataMax = 1.0;
               dotDataMin = fluidCoef;
            }
            else {
               if (normalStress == attributes.tensileStrength)
                  fluidCoef = 0.0;
               else
                  fluidCoef = attributes.fluidPressure
                        / (normalStress +  attributes.tensileStrength);

               dotData[i] = fluidCoef;

               if (fluidCoef > dotDataMax)
                  dotDataMax = fluidCoef;
               if (fluidCoef < dotDataMin)
                  dotDataMin = fluidCoef;
            }
         } else {
            if (sigma1 == sigma3) {
               dilationCoef = 0.0;
               dotData[i] = dilationCoef;
               dotDataMax = 1.0;
               dotDataMin = dilationCoef;
               // cout << "***** DILATION COEFF / 0.0 ******"
               //      << endl << flush;
               // cout << "NS " << normalStress << endl << flush;
            } else {
               dilationCoef = (sigma1-normalStress)/(sigma1-sigma3);
               dotData[i] = dilationCoef;
               if (dilationCoef > dotDataMax) dotDataMax = dilationCoef;
               if (dilationCoef < dotDataMin) dotDataMin = dilationCoef;
            }
         }
```

plotClass_findSlipV.c++

```
   fluidPressTend = attributes.fluidPressure / (slipNormalStress
                                                + attributes.tensileStrength);
```

==========

faultBuilderClass.c++

```
      XtAddCallback(attributes.closeButton, XmNactivateCallback,
                    closeButtonPress, NULL);
```

faultBuilderCB.hh

```
void closeButtonPress(Widget, XtPointer, XtPointer);
```

faultBuilderCB.c++

```
void closeButtonPress(Widget, XtPointer, XtPointer){
   faultBuilderObj.lower();
} // end function closeButtonPress
```

mohrOptionClass.c++

Added more callbacks for several scales:

```
   XtAddCallback (attributes.s3Tos1RatioScale, XmNvalueChangedCallback,
                  mohrs3Tos1RatioScaleChange, NULL);

   XtAddCallback (attributes.s2Tos1RatioScale, XmNvalueChangedCallback,
                  mohrs2Tos1RatioScaleChange, NULL);

   XtAddCallback (attributes.fluidScale, XmNvalueChangedCallback,
                  mohrChangeFluidScale, NULL);
```

mohrOptionCB.c++

Added fluid pressure and tensile strength update in the apply button callback:

```
void mohrApplyButton(Widget, XtPointer, XtPointer){
   MohrAttributeType *mattr = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();

   int sliders[3];
   int hasneg = 0;
   int smallest = 0;

   // Get values of the mohr sliders
   for (int i = 0; i < 3; i++)  {
      XmScaleGetValue(moattr->scale[i], &sliders[i]);
      if (mattr->effStress[i] < 0.0)
         hasneg = 1;
      if (sliders[i] < sliders[smallest])
         smallest = i;
   } // end for int i

   // If there is a negative value, want to shift all numbers by
   // subtracting the tensile str.  This is only when none of the
   // numers are less than the tensile str

   char buf[2000];
      sprintf(buf,
"Warning negative number entered that is\ngreater than computed tensile strength.\
nAssuming tensile strength equal to\nminimum value minus one.");

   if (hasneg == 0) {
      cmdObj.setValues(mattr->effStress[0], mattr->effStress[1],
                       mattr->effStress[2]);
      cmdObj.setFluidPress(mohrObj.fluidPressure);
      cmdObj.setTensileStr(mohrObj.tensileStr);
   } // end if
   else if (mohrObj.hasTensileStr) {
      if (mohrObj.tensileStr < mattr->effStress[smallest]){
         cmdObj.setValues(
            mattr->effStress[0] - mohrObj.tensileStr,
            mattr->effStress[1] - mohrObj.tensileStr,
            mattr->effStress[2] - mohrObj.tensileStr);
         cmdObj.setFluidPress(mohrObj.fluidPressure);
         cmdObj.setTensileStr(mohrObj.tensileStr);
      } // end if
      else
         infoWidget(moattr->mohrDialogForm, buf, mohrApplyOKCB);
   } // end else if
   else
      infoWidget(moattr->mohrDialogForm, buf, mohrApplyOKCB);


} // end of mohrApplyButton

void mohrApplyOKCB(Widget w, XtPointer, XtPointer){
   MohrAttributeType *mattr = mohrObj.getAttributes();
   MohrOptionType *moattr = mohrOptionObj.getAttributes();
```

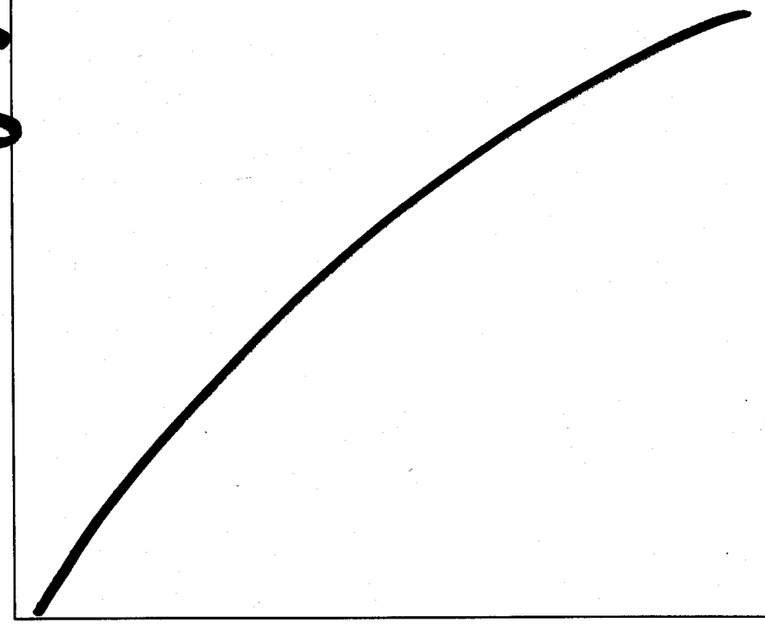


```
   int sliders[3];
   int min = 0;

   for (int i = 0; i < 3; i++)  {
      XmScaleGetValue(moattr->scale[i], &sliders[i]);

      if (sliders[i] < sliders[min])
         min = i;
   } // end for int i

   double tensile = mattr->effStress[min] - 1.0;
   cmdObj.setValues(mattr->effStress[0] - tensile,
                    mattr->effStress[1] - tensile,
                    mattr->effStress[2] - tensile);

   cmdObj.setFluidPress(mohrObj.fluidPressure);
   cmdObj.setTensileStr(mohrObj.tensileStr);

   if (w != NULL)
      XtDestroyWidget(w);
} // end of mohrApplyOKCB
```

optionClass.c++

Changed the name of the fluid / (stress+tensile) radio button:

```
fluidStr = XmStringCreateLocalized("Valve Tendency");
```

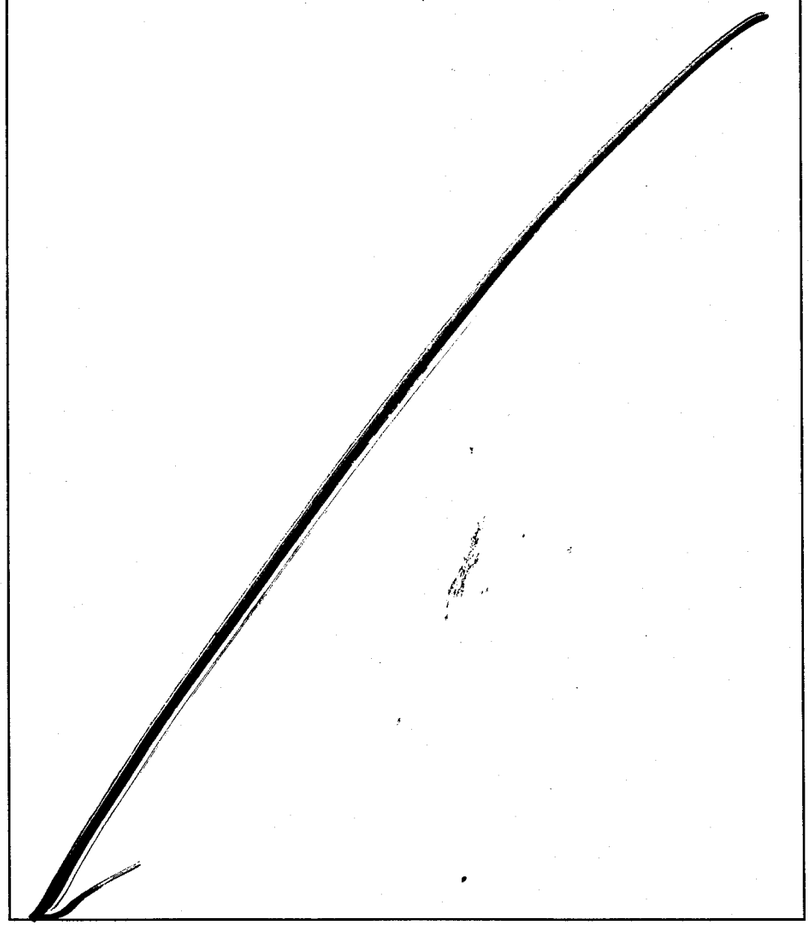


plotClass_display.c++

Toyed with the fluid to stress ratio caluculation in an attempt to discover why the spectrum of values tends to equality as tensile strength increases, but decided that this behavior is mathematically correct.

plotClass.c++

Tried to find a smoother color scale for the plotting of the fluid to stress ratio. Will continue in this same vien Tomorrow.

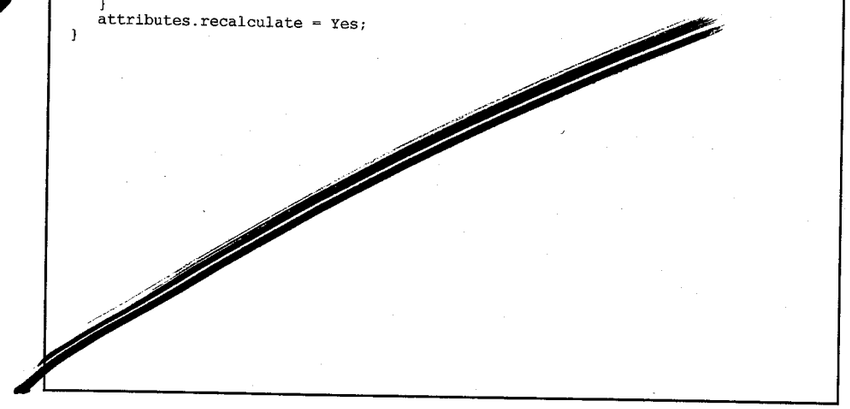


23Ju

plotClass.c++

Changed the colors to something closer to some kind of spectrum. Still not sure as to how clear this new scale is in comparison to its predecessor.

```
void  PlotClass::calcColorsFluid ( double   rad,
                    GLfloat  *red,
                    GLfloat  *grn,
                    GLfloat  *blu ) {

   double  hue = (dotDataMax-rad) / (dotDataMax-dotDataMin);
   for (int p=1; p<attributes.colorScalePwr; p++) {
      hue *= hue;
   }
   hue *= 4.0;
   if (hue < 0.0) hue = 0.0;
   double  i = floor(hue);
   double  f = hue - i;
   double  q = 1.0 - f;

   switch ((int)i) {
      case 0:
         *red = 0.0;
         *grn = 1.0;
         *blu = 0.0;
         break;
      case 1:
         *red = 0.0;
         *grn = 1.0;
         *blu = (GLfloat) q;
         break;
      case 2:
         *red = 0.0;
         *grn = (GLfloat) f;
         *blu = 1.0;
         break;
      case 3:
         *red = (GLfloat) q;
         *grn = 0.0;
         *blu = 1.0;
         break;
      case 4:
         *red = 1.0;
         *grn = 0.0;
         *blu = (GLfloat) f;
         break;
   }
   attributes.recalculate = Yes;
}
```



24Ju

mohrOptionCB.hh:

Added prototype for new initialization of citation window from a push button.

```
void mohrRockInfoButton(Widget, XtPointer, XtPointer);
```

mohrOptionCB.c++:

Added function for new initialization of citation window from a push button.

```
void mohrRockInfoButton(Widget, XtPointer, XtPointer)
{
   MohrOptionType
      * optionAttrib = mohrOptionObj.getAttributes();

   mohrRockInfoObj.initialize(optionAttrib->mohrDialogForm);
}
```

Added include for Rock Info window object.

```
#include "mohrRockInfoObj.hh"
```

mohrOptionClass.hh:

Added button for initialization of citation window to the Mohr Graph's option window.

```
Widget   rockInfoButton;
```

mohrOptionClass.c++:

Added button for initialization of citation window to the Mohr Graph's option window.

```
attributes.rockInfoButton = XtVaCreateManagedWidget("rockInfoButton",
                                 xmPushButtonWidgetClass,
                                 row3,
                                 XtVaTypedArg,
                                 XmNlabelString,
                                 XmRString, "Rock Info", 10,
                                 XmNbackground, PUSH_COLOR,
                                 NULL);

XtAddCallback (attributes.rockInfoButton, XmNactivateCallback,
               mohrRockInfoButton, NULL);
```

Makefile:

Added Rock Info files and dependencies.

```
mohrRockInfoClass.o \

mohrRockInfoClass.o:    mohrRockInfoClass.c++ mohrRockInfoClass.hh
        $(CC) $(C++FLAGS) -c $*.c++
```

viewNetGlobals.hh:

Added the MohrRockInfoClass mohrRockInfoWindow variable.

```
#include "mohrRockInfoClass.hh"

MohrRockInfoClass mohrRockInfoWindow;
```


plotClass.hh

Added variables to aid in making the arrow buttons work when held down.

```
public:
   int azButtonDown;
   int dipButtonDown;
```

gfxOptionCB.c++ :

Added while loops to functions in order to produce arrow buttons that work when held down.

```
void changeGfxDip(Widget,
                  XtPointer client_data,
                  XtPointer) {

   float
     dip, azimuth;

   plotObj.dipButtonDown = TRUE;

   while (plotObj.dipButtonDown == TRUE)
     {
       plotObj.getMouseDirPlunge(&azimuth, &dip);
       switch ((int)client_data)
         {
         case 1:
           dip++;
           if (dip > 90.0)
             dip = 90.0;
           plotObj.findSlipV(azimuth, dip);
           break;
         case 2:
           dip--;
           if (dip < 0.0)
             dip = 0.0;
           plotObj.findSlipV(azimuth, dip);
           break;
         case 3:
           plotObj.dipButtonDown = FALSE;
           break;
         default:
           break;
         }
     }
   sceneObj.update();
} // end of changeGfxDip

void changeGfxAz(Widget,
                 XtPointer client_data,
                 XtPointer) {

   float dip, azimuth;
   plotObj.azButtonDown = TRUE;

   while(plotObj.azButtonDown == TRUE)
     {
       plotObj.getMouseDirPlunge(&azimuth, &dip);
       switch ((int)client_data)
         {
         case 1:
           azimuth++;
           if (azimuth > 360.0)
             azimuth -= 360.0;
           plotObj.findSlipV(azimuth, dip);
           break;
         case 2:
           azimuth--;
           if (azimuth < 0.0)
             azimuth += 360.0;
```

```
            plotObj.findSlipV(azimuth, dip);
            break;
          case 3:
            plotObj.azButtonDown = FALSE;
            break;
          default:
            break;
          }
      }
   sceneObj.update();
} // end of changeGfxAz
```

gfxOptionClass.c++

Added call backs to make the arrow buttons stop working when released.

```
   XtAddCallback(attributes.dipRightArrow, XmNdisarmCallback,
                 changeGfxDip, (XtPointer)3);
   XtAddCallback(attributes.dipLeftArrow, XmNdisarmCallback,
                 changeGfxDip, (XtPointer)3);
   XtAddCallback(attributes.azRightArrow, XmNdisarmCallback,
                 changeGfxAz, (XtPointer)3);
   XtAddCallback(attributes.azLeftArrow, XmNdisarmCallback,
                 changeGfxAz, (XtPointer)3);
```

The above changes did not work as planned and were therefore scrapped.

Began planning for breaking Mohr Options Window up into four new windows.
There will be an option window with buttons for the other three windows, a render options window, a stress options window, and a rock options window.

mohrOptionClass.hh

```
#ifndef __MOHROPTIONCLASS_HH_
#define __MOHROPTIONCLASS_HH_

typedef struct{ // member variables for MohrOptionClass

   Widget
     mohrDialogForm,  // parent for window

     renderButton,  // render options window push button
     stressButton,  // stress options window push button
     rockButton,  // rock options window push button

     closeButton;  // close the options window
} MohrOptionType;

class MohrOptionClass{

public:

  //Constructor and Destructor
  MohrOptionClass();
  ~MohrOptionClass();

  //Public member methods
  void initialize(Widget w); //draw the window
  MohrOptionType *getAttributes(void);  //make attributes public
  void lower(void);  //close the window

private:

  //Private member variables
  MohrOptionType attributes;  //a structure of widgets
};

#endif
```

mohrOpRenderClass.hh

```
#ifndef __MOHROPRENDERCLASS_HH_
#define __MOHROPRENDERCLASS_HH_

typedef struct{ //structure to hold the attributes for MohrOpRenderClass

  Widget
    mohrRenderDialogForm,       //parent for window
    axisButton, circleButton,   //toggle buttons for visual aids
    lineWidth,                  //slider for width to draw with

    renderRadio,                //radio buttons for render mode
       renderRadio1,            //render outline of circles
       renderRadio2,            //render circles solid

    closeButton;               //exit and close window

} MohrOpRenderType;

class MohrOpRenderClass{

public:

  //Constructor and Destructor
  MohrOpRenderClass();
  ~MohrOpRenderClass();

  //Public member methods
```

```
  void initialize(Widget w);  //draw the window
  MohrOpRenderType *getAttributes(void); //make attributes public
  void lower(void); //close the window

private:

  //Private member variable
  MohrOpRenderType attributes; //a structure of widgets
};

#endif

mohrOpStressClass.hh

#ifndef __MOHROPSTRESSCLASS_HH_
#define __MOHROPSTRESSCLASS_HH_

typedef struct{ //structure to hold the attributes for MohrOpStressClass

  Widget
    mohrStressDialogForm, //parent for window

    scale[3],  //sigma 1, 2, and 3 stresses scales
    maxText,   //max value for scale[]
    minText,   //min value for scale[]

    fluidText,  // text box for typing fluid pressure
    fluidScale, // scale for specifying fluid pressure

    stressDependenceRadio,        //  radio buttons for specifying
      stressDependentRadio,       //  wether or not the stresses
      stressIndependentRadio,     //  are dependent on each other

    s3Tos1RatioText,              //  Text box for entry of ratio
    s3Tos1RatioScale,             //  Scale for specifiying Poisson's

    s2Tos1RatioText,              //  Text box for entry of ratio
    s2Tos1RatioScale,             //  Scale for specifiying Poisson's

    effText[3], //display only text box for effective stresses
    effs3Tos1RatioText, effs2Tos1RatioText,  //display only for eff. ratios

    closeButton,  // close the window
    applyButton;  // apply stress, tensile strength, and fluid press to model

  int isInfoWinOpen; // flag to tell if effective stress bounds
                     // error information window is open
                     // 0 = error window not open
                     // 1 = error window open
} MohrOpStressType;

class MohrOpStressClass{

public:

  //Constructor and Destructor
  MohrOpRockClass();
  ~MohrOpRockClass();

  //Public member methods
  void initialize(Widget w);  //draw the window
  MohrOpStressType *getAttributes(void);  //make attributes public
  void lower(void);  //close the window

private:

  //Private member variables
  MohrOptionType attributes;  //a structure of widgets
};

#endif
```

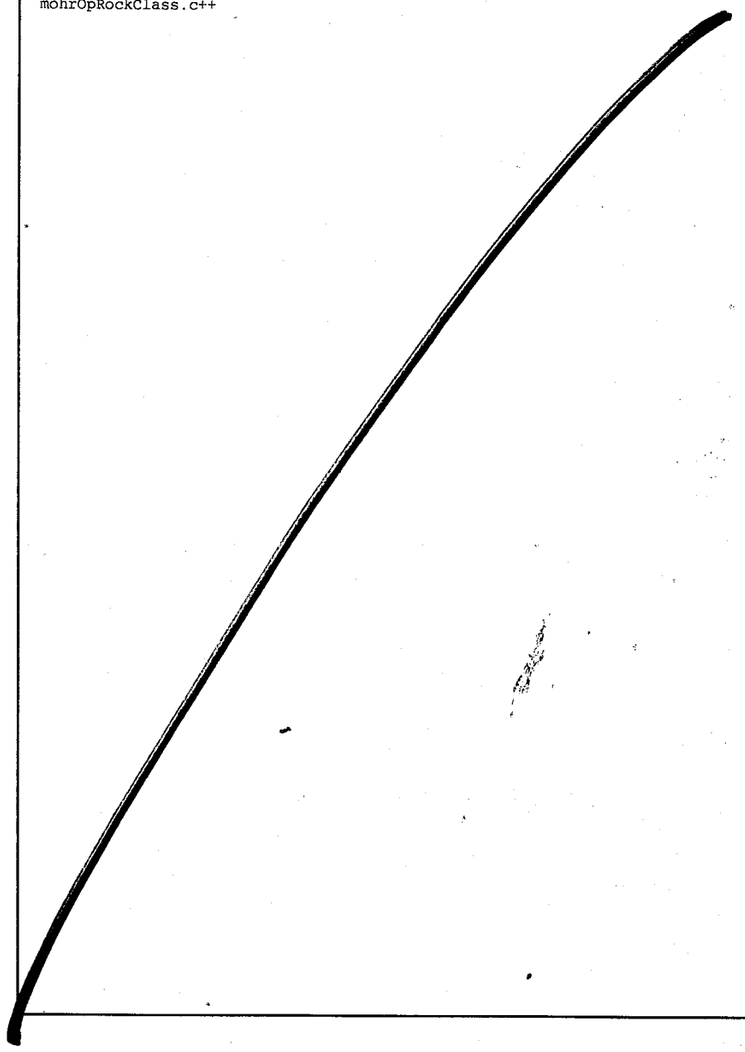


```
mohrOpRockClass.hh

#ifndef __MOHROPROCKCLASS_HH_
#define __MOHROPROCKCLASS_HH_

typedef struct{  //structure to hold the attributes for MohrOpStressClass

  Widget
    mohrRockDialogForm,  //parent for window

    rockType,
    rockMaterial,

    rockCStr[5],

    cText,  // compressional
    mText,
    sText,
    nText,  // normal stress

    closeButton,
    rockInfoButton; // push button that pops up an info window

  int rockCStrIndex[5]; // pointers to named of rocks in catagories
} MohrOpRockType;

class MohrOpRockClass{

public:

  //Constructors and Destructors
  MohrOpRockClass();
  ~MohrOpRockClass();

  //Public member methods
  void initialize(Widget w); //draw the window
  MohrOpRockType *getAttributes(void); //make attributes public
  void lower(void); //close the window

private:

  //Private member variables
  MohrOptionType attributes; //a structure of widgets
};

#endif
```

Jul 29 1997 15:29 **29July98.log** Page 1

Began editing the member methods for the new option window classes in the Mohr Graph. See the following files.

mohrOptionClass.c++
mohrOpRenderClass.c++
mohrOpStressClass.c++
mohrOpRockClass.c++



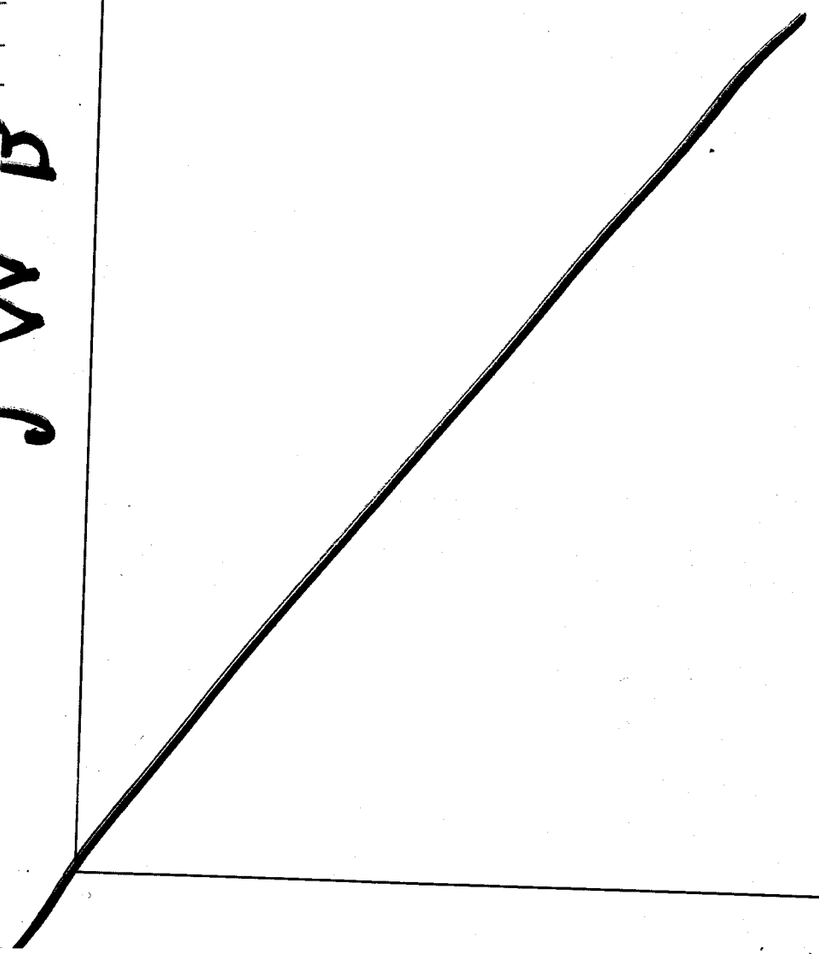
29July

Jul 30 1997 14:50 **30July97.log** Page 1

Modified many files to break the Mohr Option Window into four smaller windows.

Files modified or created:

mohrOptionClass.hh mohrOptionClass.c++
mohrOptionCB.hh mohrOptionCB.c++
mohrOpRenderClass.hh mohrOpRenderClass.c++
mohrOpRenderCB.hh mohrOpRenderCB.c++
mohrOpStressClass.hh mohrOpStressClass.c++
mohrOpStressCB.hh mohrOpStressCB.c++
mohrOpRockClass.hh mohrOpRockClass.c++
mohrOpRockCB.hh mohrOpRockCB.c++
viewNetGlobals.hh Makefile



30Ju

Fixed conflicting function call name. Changed the function

mohrOptionCB.c++

```
void mohrCloseButton(Widget, XtPointer, XtPointer){
   mohrOptionObj.lower();
} // end function mohrCloseButton
```

to the following:

mohrOpRenderClass.c++

```
   XtAddCallback (attributes.closeButton, XmNactivateCallback,
                  mohrRenderCloseButton, NULL);
```

mohrOpRenderCB.hh

```
 void mohrRenderCloseButton(Widget, XtPointer, XtPointer);
```

mohrOpRenderCB.c++

```
void mohrRenderCloseButton(Widget, XtPointer, XtPointer){
   mohrOpRenderObj.lower();
} // end function mohrCloseButton
```

mohrOpStressClass.c++

```
   XtAddCallback (attributes.closeButton, XmNactivateCallback,
                  mohrStressCloseButton, NULL);
```

mohrOpStressCB.hh

```
 void mohrStressCloseButton(Widget, XtPointer, XtPointer);
```

mohrOpStressCB.c++

```
void mohrStressCloseButton(Widget, XtPointer, XtPointer){
   mohrOpStressObj.lower();
} // end function mohrCloseButton
```

mohrOpRockClass.c++

```
   XtAddCallback (attributes.closeButton, XmNactivateCallback,
                  mohrRockCloseButton, NULL);
```

mohrOpRockCB.hh

```
 void mohrRockCloseButton(Widget, XtPointer, XtPointer);
```

mohrOpRockCB.c++

```
void mohrRockCloseButton(Widget, XtPointer, XtPointer){
   mohrOpRockObj.lower();
} // end function mohrCloseButton
```

==========

mohrOpStressCB.c++

Changed all occurences of cmdObj.setTensileStr(mohrObj.tensileStr); to cmdObj.setTensileStr(abs(mohrObj.tensileStr)); to keep within the rest of 3dstress's tensile strength bounds.

Note: must ask Dave how the tensile strength is supposed to be calculated.

Changed all occurences of cmdObj.setFluidPress(mohrObj.fluidPressure); to

```
if (mohrObj.fluidPressure > 0)
   cmdObj.setFluidPress(mohrObj.fluidPressure);
else
```



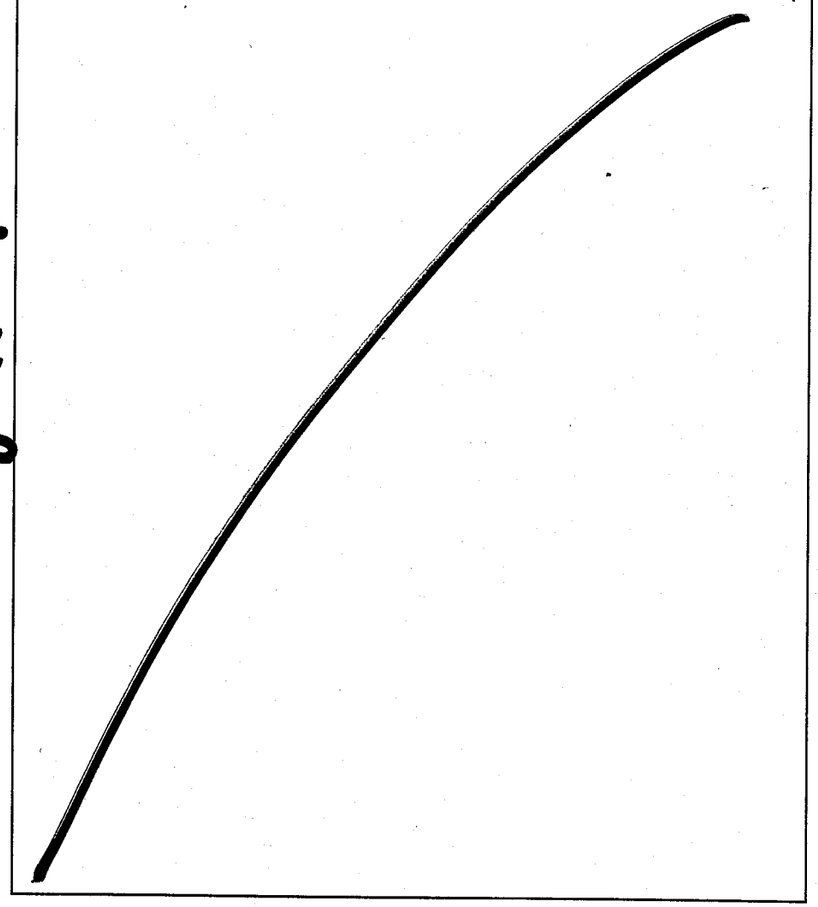
```
   cmdObj.setFluidPress(1.0);
```

to keep within the rest of 3dstress's tensile strength bounds.

==========

mohrOptionClass.c++

Played with the way the buttons in the main options window look. The goal is to eventually make them picture buttons resembling the 3dMove interface.



cmdClass.c++

Added

```
  if (value < 1)  // bounds for fp may differ
        value = 1;
```

to the function void CmdClass::setFluidPress(float value)
because the lower bound for fluid pressure in the Mohr
Graph is zero while it's one in the sliders. Also, added

```
        if (value < 0)
                value = abs(value);
```

to the function void CmdClass::setTensileStr(float value)
for the same reason as above.

==========

The previous attempt at breaking up the Mohr option window into 4 windows was scrapped in favor of a 3dmove style hide/reveal arrow. The following changes were made to implement this arrow.

mohrOptionCB.c++

Changed all occurence of

```
if (mohrObj.fluidPressure > 0)
   cmdObj.setFluidPress(mohrObj.fluidPressure);
else
   cmdObj.setFluidPress(1.0);
```

back to cmdObj.setFluidPress(mohrObj.fluidPressure); because of the increased abstraction of the above changes to void CmdClass::setFluidPress(float value).

mohrOptionClass.c++

Added the following in order to implement a 3dmove style hide/reveal arrow button.

```
#include <Xm/ArrowBG.h>

MohrOptionClass::MohrOptionClass(void) {

        // initialize variables
        attributes.arrowDir = 1;
}


attributes.arrowButton = XtVaCreateManagedWidget("arrowButton",
                            xmArrowButtonGadgetClass, row3,
                            XmNarrowDirection, XmARROW_UP,
                            XmNbackground, PUSH_COLOR, NULL);

XtAddCallback (attributes.arrowButton, XmNactivateCallback,
               mohrArrowCB, NULL);
```

Changed all occurences of rockRenderRow to attributes.rockRenderRow to facilitate arrowButton's callback.

mohrOptionClass.hh

Added

```
Widget   rockRenderRow;     //  a row to facilitate arrowButton
Widget   arrowButton;       // hides/reveales rockRenderRow
int      arrowDir;          // direction of arrow button
```

mohrOptionCB.c++

Added a new function to take care of the arrowButton.

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function: mohrArrowCB (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer, XtPointer
//
//  State Changes:  Makes 1/2 the options window dissappear/reappear
//
//  Purpose:  To conserve screen space
//
// Last Modified:  1 August 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrArrowCB(Widget, XtPointer, XtPointer){
      MohrOptionType *moattr = mohrOptionObj.getAttributes();

   moattr->arrowDir = !(moattr->arrowDir);

   if (moattr->arrowDir){
      XtManageChild(moattr->rockRenderRow);
      XtVaSetValues(moattr->arrowButton, XmNarrowDirection, XmARROW_UP, NULL);
   }
   else {
      XtUnmanageChild(moattr->rockRenderRow);
      XtVaSetValues(moattr->arrowButton, XmNarrowDirection, XmARROW_DOWN, NULL);
   }
} // end of mohrArrowCB
```

mohrOptionCB.hh

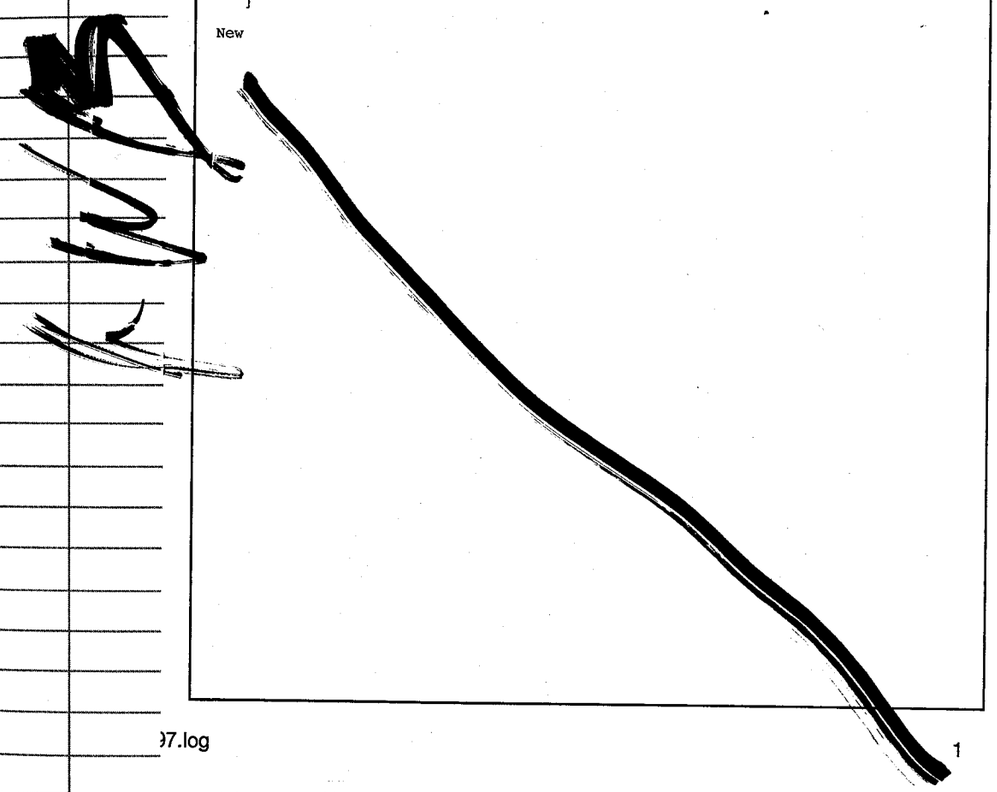
Added a new function to take care of the arrowButton.

```
void mohrArrowCB(Widget, XtPointer, XtPointer);
```

```
plotClass_display.c++

   Modified the display of Slip Az to conform with Chevron's wishes:
Changed

sprintf(strData, "Slip Az       %4.1f", (float)tmpAz);

to

if ((float)tmpAz < 100)
   sprintf(strData, "Slip Az       0%4.1f", (float)tmpAz);
else
        sprintf(strData, "Slip Az       %4.1f", (float)tmpAz);

mohrOptionClass.hh

   Added the Widget s3Tos1RatioLabel and the Widget s2Tos1RatioLabel
to the attributes structure to facilitate unmanaging them when in
independent stress mode.

mohrOptionClass.c++

   Changed all occurances of s3Tos1RatioLabel and s2Tos1RatioLabel to
attributes.s3Tos1RatioLabel and attributes.s2Tos1RatioLabel because of
the above change in mohrOptionClass.hh.

   Removed all XtSetSensitive calls on the dependent stress
features since the current plan is to simply unmanage them.

   Added to the bottom the following lines to make sure the dependent
stress stuff is initially hidden:

   XtUnmanageChild(optionAttrib->s3Tos1RatioLabel);
   XtUnmanageChild(optionAttrib->s2Tos1RatioLabel);
   XtUnmanageChild(optionAttrib->s3Tos1RatioText);
   XtUnmanageChild(optionAttrib->s2Tos1RatioText);
   XtUnmanageChild(optionAttrib->s3Tos1RatioScale);
   XtUnmanageChild(optionAttrib->s2Tos1RatioScale);

   Removed "Effective " from the lables on the effective ratios to cut
down on window size.

   Changed the order of the sigma slider scales and the radio button and
sliders for dependent/independent stress.

mohrOptionCB.c++

   In the function mohrToggleStressMode, changed the XtSetSensitive calls
to XtManageChild and XtUnmanageChild calls.

=========================================================================

mohrOptionClass.hh

   Added Widget colStress2 to facilitate not drawing stress dependent
widgets at initial start up of options window.  Removed ratio lables.

mohrOptionClass.c++

   Implemented the colStress2 widget by linking the stress dependent
widgets to it and removing the final unmanaging of these widgets.  Changed
attributes.s?Tos1RatioLabel to Widget s?Tos1RatioLabel.

mohrOptionCB.c++

   Changed the function mohrToggleStress mode in the following way:

Previous:

   if (mohrAttrib->stressMode == 1)
   {
```



```
      XtUnmanageChild(optionAttrib->s3Tos1RatioLabel);
      XtUnmanageChild(optionAttrib->s2Tos1RatioLabel);
      XtUnmanageChild(optionAttrib->s3Tos1RatioText);
      XtUnmanageChild(optionAttrib->s2Tos1RatioText);
      XtUnmanageChild(optionAttrib->s3Tos1RatioScale);
      XtUnmanageChild(optionAttrib->s2Tos1RatioScale);
   }
   else  // if stress is dependent, activate the ratio scale
   {
      findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3 );
      recalcSigmas(min, mid, max, mohrAttrib->sigmas[max]);

      XtManageChild(optionAttrib->s3Tos1RatioLabel);
      XtManageChild(optionAttrib->s2Tos1RatioLabel);
      XtManageChild(optionAttrib->s3Tos1RatioText);
      XtManageChild(optionAttrib->s2Tos1RatioText);
      XtManageChild(optionAttrib->s3Tos1RatioScale);
      XtManageChild(optionAttrib->s2Tos1RatioScale);

      XmScaleSetValue(optionAttrib->scale[0], mohrAttrib->sigmas[0] * 1000);
      XmScaleSetValue(optionAttrib->scale[1], mohrAttrib->sigmas[1] * 1000);
      XmScaleSetValue(optionAttrib->scale[2], mohrAttrib->sigmas[2] * 1000);

      updateEffective();
      mohrObj.failure();
      mohrObj.display();
   }

New
```



Attempted to fix problem with both rows dissapearing for the dependent stress controlls on the Mohr option window:

mohrOptionClass.hh

Added the following:

```
Widget   rowStress,   //  row for unmanaging dstress specific stuff
         colStress2, //  col for unmanaging dstress specific stuff
```

mohrOptionClass.c++

Modified previous to the following:

```
attributes.rowStress = XtVaCreateWidget("rowStress",
                       xmRowColumnWidgetClass,
                       subColRow1,
                       XmNorientation, XmHORIZONTAL,
                       XmNnumColumns, 2,
                       XmNpacking, XmPACK_TIGHT,
                       XmNbackground, WIDGET_COLOR,
                       NULL);

frame = XtVaCreateManagedWidget("frameStress", xmFrameWidgetClass,
                                attributes.rowStress,
                                XmNbackground, WIDGET_COLOR,
                                NULL);

attributes.colStress2 = XtVaCreateWidget("attributes.colStress2Widget",
                        xmRowColumnWidgetClass, frame,
                        XmNorientation, XmVERTICAL,
                        XmNnumColumns, 7,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR, NULL);

char strHolder[100]; // holder to initialize text box with number

strOne = motifFontObj.create("", "s", "3");  // sigma 3
strTwo = motifFontObj.create(" to ", "s", "1 Ratio"); // to sigma 1 ratio
strFinal = XmStringConcat(strOne, strTwo); // combine above two strings
XmStringFree(strOne);    // free first string

// lable for s3Tos1ratio text box and scale
Widget s3Tos1RatioLabel = XtVaCreateManagedWidget("s3Tos1RatioLabel",
                          xmLabelWidgetClass, attributes.colStress2,
                          XmNlabelString, strFinal,
                          XmNfontList, motifFontObj.fontListing,
                          XmNbackground, WIDGET_COLOR,
                          NULL);
// free the title string
XmStringFree(strFinal);

// create the text box
attributes.s3Tos1RatioText = XtVaCreateManagedWidget("ratio1Text",
                             xmTextFieldWidgetClass, attributes.colStress2,
                             XmNtraversalOn, True,
                             XmNcolumns, 10,
                             XmNbackground, FILL_COLOR,
                             NULL);

// initialize the text box's value
sprintf(strHolder, "%lf", mattr->s3Tos1Ratio);
XmTextSetString(attributes.s3Tos1RatioText, strHolder);

// define a scale for the ratio between dependent stresses
attributes.s3Tos1RatioScale = XtVaCreateManagedWidget("ratio1Scale",
                              xmScaleWidgetClass, attributes.colStress2,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, mattr->stressScaleMax,
                              XmNminimum, mattr->stressScaleMin,
                              XmNvalue, (int) (mattr->s3Tos1Ratio * 100),
```



```
                              XmNscaleMultiple, 1,
                              XmNdecimalPoints, 2,
                              XmNshowValue, False,
                              XmNbackground, WIDGET_COLOR,  NULL);

strOne = motifFontObj.create("", "s", "2"); // sigma 2
strFinal = XmStringConcat(strOne, strTwo);  // concatenate two strings
XmStringFree(strOne);   // free the two extra stings, holders
XmStringFree(strTwo);

// lable for s2Tos1ratio text box and scale
Widget s2Tos1RatioLabel = XtVaCreateManagedWidget("s2Tos1RatioLabel",
                          xmLabelWidgetClass, attributes.colStress2,
                          XmNlabelString, strFinal,
                          XmNfontList, motifFontObj.fontListing,
                          XmNbackground, WIDGET_COLOR,
                          NULL);
// free the title string
XmStringFree(strFinal);

// create a text box
attributes.s2Tos1RatioText = XtVaCreateManagedWidget("ratio2Text",
                             xmTextFieldWidgetClass, attributes.colStress2,
                             XmNtraversalOn, True,
                             XmNcolumns, 10,
                             XmNbackground, FILL_COLOR,
                             NULL);

// initialize the new text box with a number
sprintf(strHolder, "%lf", mattr->s2Tos1Ratio);
XmTextSetString(attributes.s2Tos1RatioText, strHolder);

attributes.s2Tos1RatioScale = XtVaCreateManagedWidget("ratio2Scale",
                              xmScaleWidgetClass, attributes.colStress2,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, mattr->stressScaleMax,
                              XmNminimum, mattr->stressScaleMin,
                              XmNvalue, (int) (mattr->s2Tos1Ratio * 100),
                              XmNscaleMultiple, 1,
                              XmNdecimalPoints, 2,
                              XmNshowValue, False,
                              XmNbackground, WIDGET_COLOR,  NULL);
  // end of define a text and scale for dependent stresses ratios
```

mohrOptionCB.c++

Modified the following function:

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//  Function:  mohrToggleStressMode (not a member function)
//
//  File:  mohrOptionCB.c++
//
//  Arguments:  Widget, XtPointer clientData, XtPointer callData
//              clientData is the data that represents the action the
//              user took.  Ask Robert Boenau what the hell the rest are.
//
//  State Changes:  mohrObj.attributes->stressMode, changes the stress
//                  mode between dependent stresses and independent
//                  stresses. When stress is set to independent, the stress
//                  dependency ratio scale is greyed out;  when it is set to
//                  dependent, the stress dependency ratio is activated.
//                  When stresses are set to dependent, sigmas are recalculated
//                  to take ratios into account.
//
//  Purpose:  This function switches between the two stress modes
//
// Last Modified:  14 July 1997, Joshua Buckner (jbuckner@cs.trinity.edu)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void mohrToggleStressMode(Widget, XtPointer clientData, XtPointer)
{
```

```
   // get the attributes of the current Mohr graph for changing
   MohrAttributeType *mohrAttrib = mohrObj.getAttributes();

   // get the attributes of the option window (widgets and such)
   MohrOptionType *optionAttrib = mohrOptionObj.getAttributes();

   // set the stress mode to the appropriate value (0 or 1)
   // stressMode will be 0 for dependent stresses and 1 for independent
   mohrAttrib->stressMode = (int)clientData;

   int
      min, mid, max;

   // if stress is independent, grey out the ratio scale
   if (mohrAttrib->stressMode == 1)
   {
      XtUnmanageChild(optionAttrib->rowStress);
   }
   else  // if stress is dependent, activate the ratio scale
   {
      findMinMidMax( & min, & mid, & max, mohrAttrib->sigmas, 3 );
      recalcSigmas(min, mid, max, mohrAttrib->sigmas[max]);

      XtManageChild(optionAttrib->colStress2);
      XtManageChild(optionAttrib->rowStress);

      XmScaleSetValue(optionAttrib->scale[0], mohrAttrib->sigmas[0] * 1000);
      XmScaleSetValue(optionAttrib->scale[1], mohrAttrib->sigmas[1] * 1000);
      XmScaleSetValue(optionAttrib->scale[2], mohrAttrib->sigmas[2] * 1000);

      updateEffective();
      mohrObj.failure();
      mohrObj.display();
   }
} // end function mohrToggleStressMode
```

---

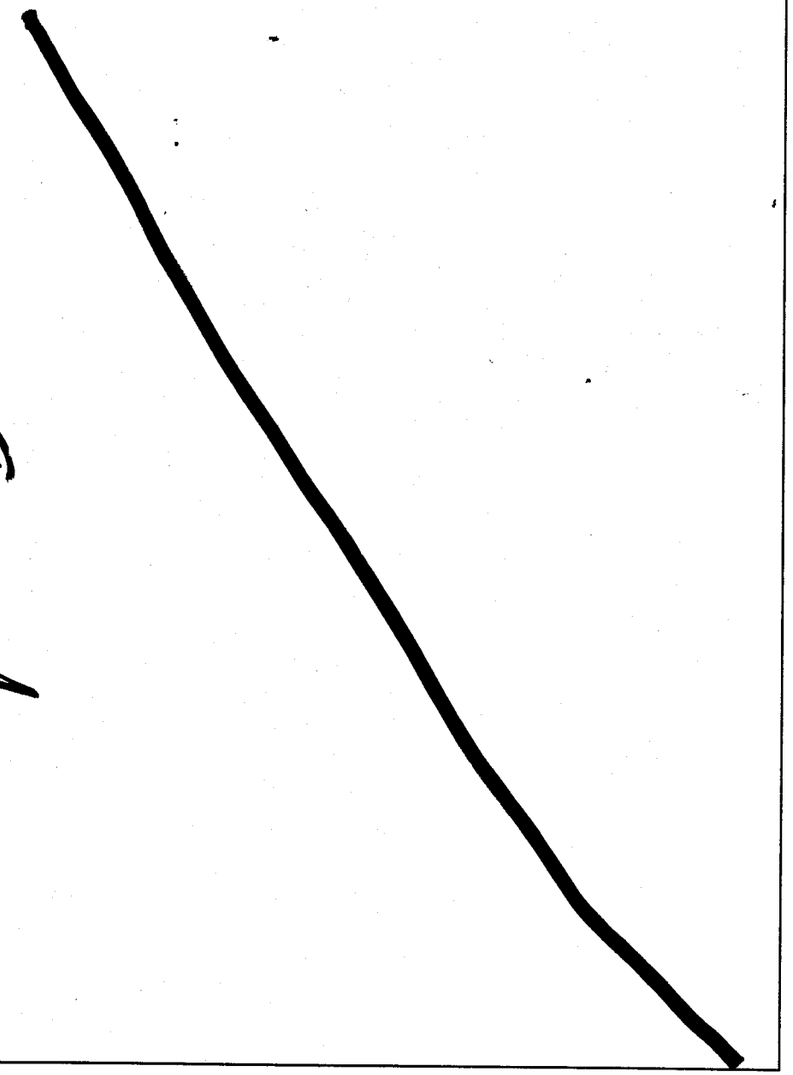
mohrCallbacks.c++

Added Fluid Pressure to the display window by modifying mohrCBdraw() as follows:

```
   if (mattr->showText) {
      // Should make it so that when the labels are close to eachother,
      // they wont collide

      motifFontObj.drawGLSymbol(1.0, 1.0, 0.0, sigmas[s1], 0.0, "s1");
      motifFontObj.drawGLSymbol(1.0, 1.0, 0.0, sigmas[s2], 0.0, "s2");
      motifFontObj.drawGLSymbol(1.0, 1.0, 0.0, sigmas[s3], 0.0, "s3");

      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0.0, 20.0, 0.0, 20.0);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();

      char str1[200];

      sprintf(str1,"Fluid Pressure: %lf", mohrObj.fluidPressure);
      motifFontObj.drawGLFont(0.0, 1.0, 1.0, 0.5, 19.0, str1);

      if (mohrObj.hasTensileStr == 1) {
         sprintf(str1,"Tensile Strength: %lf", mohrObj.tensileStr);
         motifFontObj.drawGLFont(0.0, 1.0, 1.0, 0.5, 18.0, str1);
      } // end if
      else
         motifFontObj.drawGLFont(0.0, 1.0, 1.0, 0.5, 18.0,
                                 "Tensile Strength: undefined");
      if (mohrObj.hasFailed == 1)
         motifFontObj.drawGLFont(0.0, 1.0, 1.0, 0.5, 17.0,
```

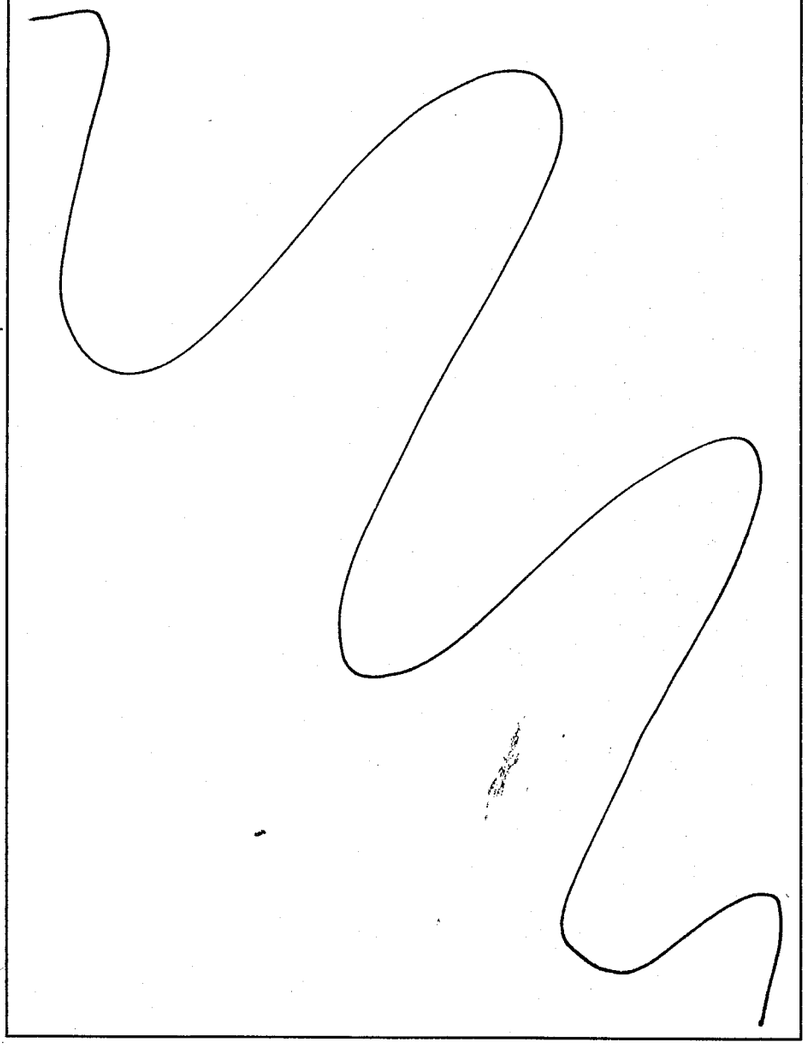


```
                                 "Failure has occurred");
   } // end if mattr->showText
```



Aug 6 1997 15:56 **6August97.log** Page 1

Gave demo on performer with the new demo account, and realized that the V2.0 Alpha binary for 3dstress was outdated. Replaced binary on performer's demo account to facilitate use of apply button in the Mohr Graph window and correctly working Leakage Factor plotting.



JWB

Aug 7 1997 15:57 **SciNote/7August97.log** Page 1

cmdClass_toggleCB.c++

Included fluid pressure and tensile strength in the save and load routines for the slider window with the following modifications.

```
#include "cmdObj.hh"      // to call setFluidPress and setTensileStr
#include "plotObj.hh"     // to allowing saving and loading of
                          // fluid pressure and tensile strength

void saveMag(Widget w, XtPointer, XtPointer call_data){
   PlotAttrType
      plotAttr = plotObj.getAttributes();
   .
   .
   .
   fprintf(fid,  "%f %f\n",  plotAttr.fluidPressure,      // save fluid
                             plotAttr.tensileStrength);   // save tensile
   fprintf(fid, "%i\n", firstChoice);
   fclose(fid);
   cmdSaveWidget.lower();
}

void loadMag(Widget w, XtPointer, XtPointer call_data){
   char            *fileName = (char *)XtMalloc(sizeof(char) * 100);
   .
   .
   .
   float                    newFluid,  newTensile;
   if (fileName == NULL) {
      fprintf(stderr,"Fatal error: unable to perform malloc");
      exit(41);
   }
   .
   .
   .
   // Start reading file
   int count = 1;
   int values[10];

   char buf[400];
   // X Y Z
   if (fscanf(fp, "%i %i %i", &values[0], &values[1], &values[2]) != 3) {
      sprintf(buf,"Invalid Magnitude File Line %i", count);
      infoWidget(w, buf);
      return;
   }
   if (values[0] < 0 || values[0] > 360 ||
       values[1] < 0 || values[1] > 360 ||
       values[2] < 0 || values[2] > 360){
      sprintf(buf,"Invalid Magnitude File Line %i", count);
      infoWidget(w, buf);
      return;
   }
   count++;
   .
   .
   .
   // fluid pressure and tensile strength
   if (fscanf(fp, "%f %f", &newFluid,  &newTensile) != 2) {
      sprintf(buf,"Invalid Magnitude File Line %i", count);
      infoWidget(w, buf);
```

JWB


```
    return;
}
if ( (newFluid < .001 || newFluid > 100) &&
     (newTensile < .001 || newTensile > 100) ){
   sprintf(buf,"Invalid Magnitude File Line %i", count);
   infoWidget(w, buf);
   return;
}
count++;

// firstChoice
if (fscanf(fp, "%i", &values[9]) != 1) {
   sprintf(buf,"Invalid Magnitude File Line %i", count);
   infoWidget(w, buf);
   return;
}
if (values[9] < 1 || values[9] > 3) {
   sprintf(buf,"Invalid Magnitude File Line %i", count);
   infoWidget(w, buf);
   return;
}

XmScaleSetValue(scaleX, values[0]);
plotObj.setStressX(values[0]);
XmScaleSetValue(scaleY, values[1]);
plotObj.setStressY(values[1]);
XmScaleSetValue(scaleZ, values[2]);
plotObj.setStressZ(values[2]);

setFluidPress(newFluid);
setTensileStr(newTensile);

   .
   .
   .

   sceneObj.update();
   fclose (fp);

}
```

---

cmdClass.c++

Changed the tensile strength scale's upper bound in the sliders window to match the Mohr graph.

```
   tensileStrW = XtVaCreateManagedWidget(
                                         "TensileStr",
                                         xmScaleWidgetClass,
                                         colWidget2,
                                         XtVaTypedArg,
                                         XmNtitleString,
                                         XmRString, "Tensile Strength", 17,
                                         XmNorientation, XmHORIZONTAL,
                                         XmNmaximum, 40000,
                                         XmNminimum, 0,
                                         XmNscaleMultiple, 1,
                                         XmNvalue, 10,
                                         XmNscaleHeight, 20,
                                         //XmNscaleWidth, 150,
                                         XmNdecimalPoints, 3,
                                         XmNshowValue, True,
                                         XmNbackground, WIDGET_COLOR,
                                         NULL);
```

Modified the functions setFluidPress and setTensileStr so that the value passed to the respective scale is of the correct order of magnitude.

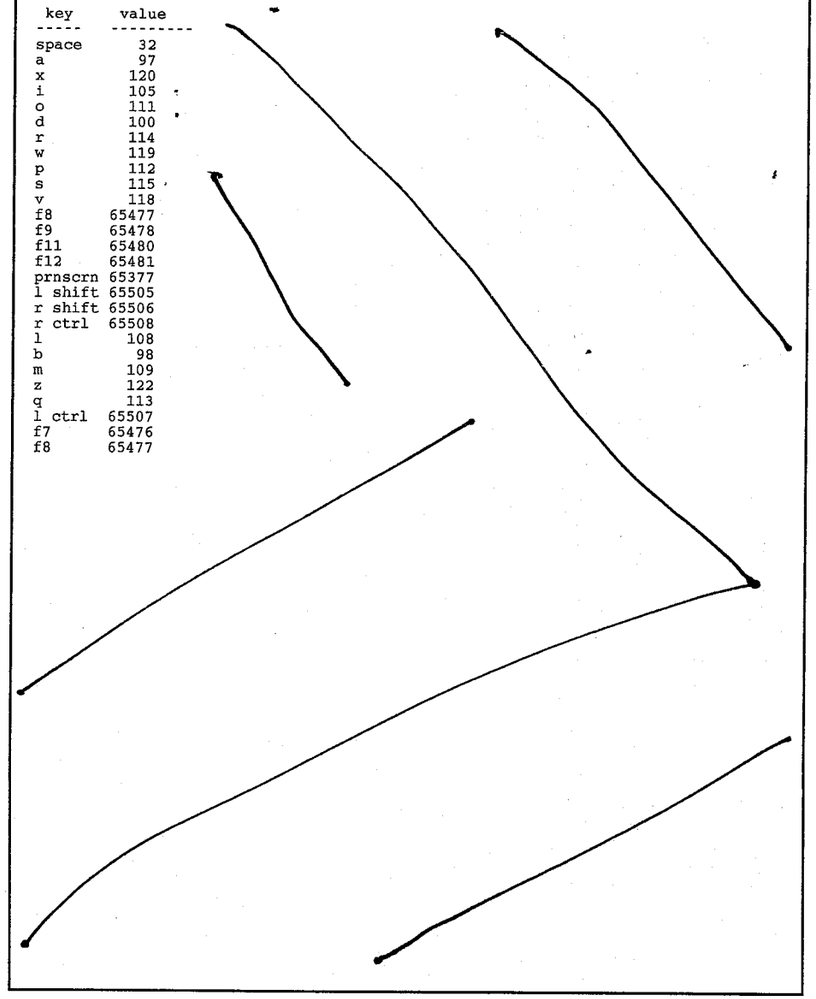
```
   XmScaleSetValue(fluidPressW, value * 1000);
```


JWB


```
and
   XmScaleSetValue(tensileStrW, value * 1000);
```

Looked at mohrCallbacks to find out what adding quick keyboard commands would entail. Some numerical values for certain key presses

key	value
space	32
a	97
x	120
i	105
o	111
d	100
r	114
w	119
p	112
s	115
v	118
f8	65477
f9	65478
f11	65480
f12	65481
prnscrn	65377
l shift	65505
r shift	65506
r ctrl	65508
l	108
b	98
m	109
z	122
q	113
l ctrl	65507
f7	65476
f8	65477



mohrCallbacks.c++

Updated the mohrCBinput function to handle many more keyboard commands.

```
switch (event->type) {
   case KeyPress:
      length = XLookupString((XKeyEvent *)event,
                             buffer,
                             9,
                             &keysym,
                             &composeStatus
                            );

      if ((length > 0) && (length <=9)) buffer[length]='\0';
      kstring = kstringA;
      kstring = XKeysymToString(keysym);
      switch ((int)keysym) {
         case  97:            //  "a" key -- toggle graph's axes
         case 65477: // f8
            if (mattr->showAxis == 0)    // if axes not displayed
            {
               mattr->showAxis = 1;      // show axes

               // if there is an option window, update the button
               if (moattr->mohrDialogForm != NULL)
                  XmToggleButtonSetState(moattr->axisButton, TRUE, FALSE);

            }
            else
            {
               mattr->showAxis = 0;      // hide axes

               if (moattr->mohrDialogForm != NULL)
                  XmToggleButtonSetState(moattr->axisButton, FALSE, FALSE);

            }

            mohrObj.display();           // update screen
            break;
      case 105:            //  "i" key -- zoom in
         mattr->viewDistance -= mattr->viewDistance * 0.01;
         if (mattr->viewDistance < 0.1)
            mattr->viewDistance = 0.1;
         mohrObj.display();
         break;
      case 111:            //  "o" key -- zoom out
         mattr->viewDistance += mattr->viewDistance * 0.1;
         mohrObj.display();
         break;
      case 114:            //  "r" key -- reset view
         mattr->viewDistance = 120.0;
         for (int i = 0; i < 3; i++)
            mattr->center[i] = 0.0;    // recenter the graph
         mohrObj.display();            // initiate changes
      case 120:            //  "x" key
         break;
      case 65480:   // f11 -- hide or reveal inner circles
         if (mattr->showInnerCircles == 0)    // if axes not displayed
         {
            mattr->showInnerCircles = 1;      // show axes

            // if there is an option window, update the button
            if (moattr->mohrDialogForm != NULL)
               XmToggleButtonSetState(moattr->circleButton, TRUE, FALSE);

         }
         else
         {
            mattr->showInnerCircles = 0;      // hide axes

            if (moattr->mohrDialogForm != NULL)
               XmToggleButtonSetState(moattr->circleButton, FALSE, FALSE);

         }
         mohrObj.display();
         break;
      case 65481:   // f12 -- line or solid mode toggle
         if (mattr->render == 1)
         {
            mattr->render = 2;

            // if there is an option window, update the button
            if (moattr->mohrDialogForm != NULL)
            {
               XmToggleButtonSetState(moattr->renderRadio1, TRUE, FALSE);
               XmToggleButtonSetState(moattr->renderRadio2, FALSE, FALSE);
            }
         }
         else
         {
            mattr->render = 1;

            // if there is an option window, update the button
            if (moattr->mohrDialogForm != NULL)
            {
               XmToggleButtonSetState(moattr->renderRadio2, TRUE, FALSE);
               XmToggleButtonSetState(moattr->renderRadio1, FALSE, FALSE);
            }
         }
         mohrObj.display();
         break;
      case 65377: // printscreen -- save the graph
         saveWindow(mattr->mohrTop);
         break;
      } // end switch
```

helpWidget.c++

Added valid mohr keyboard inputs to its quick-help list:

```
char *help7[] = {
"Mohr Quick Help",
"",
"Mouse:",
"    <left button> Change position of graph",
"    <middle button> Zoom in or out",
"    <right button> Change log mode",
"",
"Keyboard:",
"    (a) Toggle showing of axis",
"    (i) Zoom in",
"    (o) Zoom out",
"    (r) Reset viewer",
"    (Spacebar) Turn selection on",
"    (F8) Toggle showing of axis",
"    (F11) Toggle showing of inner circles",
"    (F12) Change render mode",
"    (Print Screen) Create rgb image of window",
""
};

.
.
.

      case 7: // Mohr
         lineCnt = 18;
```

Attempted to create a toggle to display/hide bounding box.

mapCallbacks.c++

Modified mapCBdraw so that display of the bounding box is optional:

```
int      mapShowBoundingBox = TRUE;  // global variable

if(mapShowBoundingBox) // global variable declared in mapCallbacks.c++
   mapDrawBorder();
```

mapOptionClass.hh

Added a widget to the MapOptionType struct:

```
Widget   boundingBoxToggle;  // hide reveal bounding box of map
```

mapOptionClass.c++

```
#include "mapCallbacks.c++"     // to access mapShowBoundingBox boolean

extern int mapShowBoundingBox;   // from mapCallbacks.c++

in function void MapOptionClass::initialize(Widget w)

   Widget     frame,
              colWidget,
              colToggle,
              row1,
              row2,
              col1,
              col2,
              col3,
              col4;

   frame = XtVaCreateManagedWidget("frame", xmFrameWidgetClass,
              colWidget,
              XmNbackground, WIDGET_COLOR, NULL);

   colToggle  = XtVaCreateWidget("colToggle",
                        xmRowColumnWidgetClass,
                        frame,
                        XmNorientation, XmVERTICAL,
                        XmNnumColumns, 5,
                        XmNpacking, XmPACK_TIGHT,
                        XmNbackground, WIDGET_COLOR,
                        NULL);

   attributes.boundingBoxToggle = XtVaCreateManagedWidget("axis",
                                    xmToggleButtonGadgetClass,
                                    col1,
                                    XtVaTypedArg,
                                    XmNlabelString,
                                    XmRString,"Show Bounding Box",18,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

   if (mapShowBoundingBox)
      XmToggleButtonSetState(attributes.axisButton, TRUE, TRUE);

   XtAddCallback (attributes.boundingBoxToggle, XmNvalueChangedCallback,
                  mapBoundingBoxToggleChange, (XtPointer)0);

   XtManageChild(colToggle);
```

mapOptionCB.hh

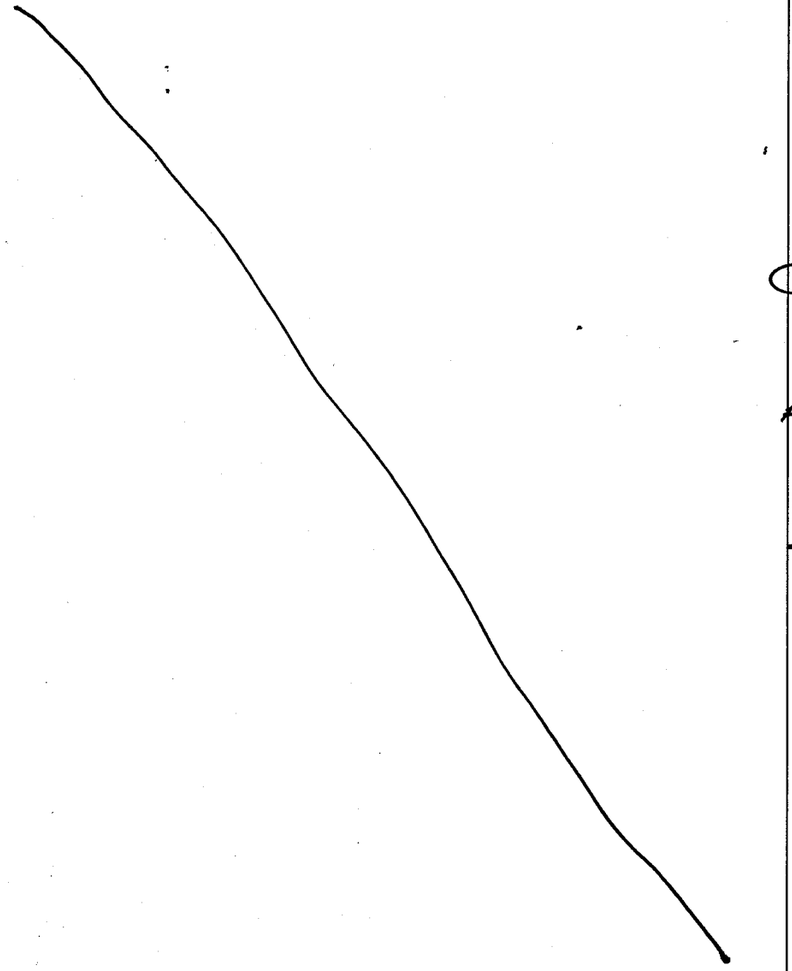
```
void mapBoundingBoxToggleChange (Widget, XtPointer, XtPointer);
```

mapOptionCB.c++

```
#include "mapCallBacks.hh"  // for mapShowBoundingBox
```



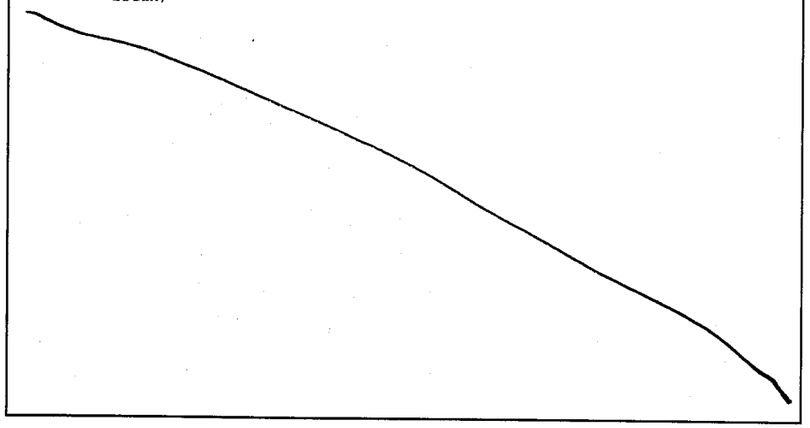
```
extern int   mapShowBoundingBox;  // from mapCallBacks.hh

void mapBoundingBoxToggleChange (Widget, XtPointer, XtPointer)
{
   mapShowBoundingBox = ! mapShowBoundingBox;  // set to oposite
   mohrObj.display();
} // end function mohrToggleButtonChange
```



```
   Must rethink structure of map bounding box toggle as it is causing
linker warning.

mapOptionClass.c++

   Removed the following lines:

   #include "mapCallbacks.c++"  // to access mapShowBoundingBox boolean
   extern int mapShowBoundingBox;   // from mapCallbacks.c++

   Added or modified lines to appear as follows.

   #include "mapObj.hh"         // to access mapShowBoundingBox
   MapAttrType     mapAttr = mapObj.getAttributes(); // to access mapShowBoundingB
ox
   if (mapAttr.mapShowBoundingBox == YesM)
      XmToggleButtonSetState(attributes.boundingBoxToggle, TRUE, TRUE);

mapOptionCB.c++

   Removed the following lines:

   #include "mapCallbacks.c++"  // to access mapShowBoundingBox boolean
   extern int mapShowBoundingBox;   // from mapCallbacks.c++

   Added or modified lines to appear as follows.

void mapBoundingBoxToggleChange (Widget, XtPointer, XtPointer)
{
   MapAttributeType *mattr = mapObj.getAttributes();

   if (mapShowBoundingBox == YesM)
      mapShowBoundingBox = NoM;
   else
      mapShowBoundingBox = YesM;

   mapObj.display();
} // end function mohrToggleButtonChange

mapClass.hh

   Added the following line to the attributes structure:

   MapYesNoType    mapShowBoundingBox;

mapCallbacks.c++

   Removed

   int      mapShowBoundingBox = TRUE;

   Modified to the following in the function mapCBdraw.

   if(mattr->mapShowBoundingBox == YesM)
      mapDrawBorder();

   Modified the following in or for the function mapCBinput.

   #include <Xm/ToggleB.h>           // for mapCBinput keyboard stuff
   #include "mapOptionObj.hh"

   MapOptionType *moattr = mapOptionObj.getAttributes();

            case 98:              //  "b" key
               if (mattr->mapShowBoundingBox == NoM)
               {  // if bounding box not displayed
                  mattr->mapShowBoundingBox = YesM;      // show bounding box

                  // if there is an option window, update the button
                  if (moattr->mapDialogForm != NULL)
                     XmToggleButtonSetState(moattr->boundingBoxToggle, TRUE, FALSE
```



```
);
                  }
                  else
                  {
                     mattr->mapShowBoundingBox = NoM;      // hide bounding box

                     if (moattr->mapDialogForm != NULL)
                        XmToggleButtonSetState(moattr->boundingBoxToggle, FALSE, FALS
E);
                  }
                  mapObj.display();           // update screen
                  break;

helpWidget.c++

   Added the following for displaying/hiding the bounding box in the
map tool:

char *help3[] = {
"Map Viewer Quick Help",
"",
"Mouse:",
"    <left button> Change position of map",
"    <middle button> Zoom in or out",
"    <right button> Select fault",
"",
"Keyboard:",
"    (a) Toggle showing of bounding area",
"    (o) Zoom out",
"    (i) Zoom in",
"    (r) Reset viewer",
"    (Print Screen) Create rgb image of window",
""
};

      case 3: // Map
         lineCnt = 14;
         strList = (XmStringTable) XtMalloc(lineCnt * sizeof(XmString));
         if (strList == NULL) {
            fprintf(stderr,"Fatal error: unable to perform malloc");
            exit(8);
         }
         for ( i = 0;  i < lineCnt; i++)
            strList[i] = XmStringCreateLocalized(help3[i]);
         break;
```



Added a toggle to display/hide the bounding box in the 3dViewer.

viewerOptionClass.hh

Added the following to the attribute structure.

```
Widget boundingBoxToggle;
```

viewerOptionClass.c++

In function initialize:

```
attributes.boundingBoxToggle = XtVaCreateManagedWidget("axis",
                                    xmToggleButtonGadgetClass,
                                    col3,
                                    XtVaTypedArg,
                                    XmNlabelString,
                                    XmRString,"Show Bounding Box",18,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

if (vattr->viewerShowBoundingBox == YesV)
   XmToggleButtonSetState(attributes.boundingBoxToggle, TRUE, TRUE);

XtAddCallback (attributes.boundingBoxToggle, XmNvalueChangedCallback,
               viewerBoundingBoxToggleChange, (XtPointer)0);
```

viewerOptionCB.hh

Added the following prototype.

```
void viewerBoundingBoxToggleChange (Widget, XtPointer, XtPointer);
```

viewerOptionCB.c++

Added or modified lines to appear as follows.

```
void viewerBoundingBoxToggleChange (Widget, XtPointer, XtPointer)
{
   ViewerAttributeType *vattr = viewerObj.getAttributes();

   if (vattr->viewerShowBoundingBox == YesV)
      vattr->viewerShowBoundingBox = NoV;
   else
      vattr->viewerShowBoundingBox = YesV;

   viewerObj.display();
} // end function mohrToggleButtonChange
```

viewerClass.hh

Added the following line to the attributes structure:

```
   ViewerYesNoType    viewerShowBoundingBox;
```

viewerCallbacks.c++

Modified to the following in the function viewerCBdraw.

```
      if(vattr->viewerShowBoundingBox == YesV)
      {
         drawBoundingBox(vattr->min, vattr->max,
                         vattr->viewDistance, (int)selecting, mousePrevX,
                         mousePrevY, aspect, vattr->viewDistance);
      }
```

Modified the following in or for the function viewerCBinput.

```
   viewerOptionType *voattr = viewerOptionObj.getAttributes();
```



```
            case 98:              //  "b" key
               if (vattr->viewerShowBoundingBox == NoV)
               {  // if bounding box not displayed
                  vattr->viewerShowBoundingBox = YesV;       // show bounding box

                  // if there is an option window, update the button
                  if (voattr->viewerDialogForm != NULL)
                     XmToggleButtonSetState(voattr->boundingBoxToggle, TRUE, FALSE
);
               }
               else
               {
                  mattr->viewerShowBoundingBox = NoV;       // hide bounding box

                  if (voattr->viewerDialogForm != NULL)
                     XmToggleButtonSetState(voattr->boundingBoxToggle, FALSE, FALS
E);
               }
               viewerObj.display();          // update screen
               break;
```

helpWidget.c++

Added the following for displaying/hiding the bounding box in the viewer tool:

```
char *help2[] = {
"Surface Viewer Quick Help",
"",
"Mouse:",
"    <left button> Nothing",
"    <middle button> Zoom in or out",
"    <right button> Rotate the viewer",
"",
"Keyboard:",
"    (b) Toggle showing of bounding box",
"    (o) Zoom out",
"    (i) Zoom in",
"    (r) Reset viewer",
"    (z) Increase vertical exaggeration",
"    (q) Decrease vertical exaggeration",
"    (F7) Toggle showing of axis",
"    (F8) Toggle showing of bounding box",
"    (F9) Toggle showing of base",
"    (F11) Toggle showing of points",
"    (F12) Change render mode",
"    (Print Screen) Create rgb image of window",
""
};

      case 2: // Viewer
         lineCnt = 19;
         strList = (XmStringTable) XtMalloc(lineCnt * sizeof(XmString));
         if (strList == NULL) {
            fprintf(stderr,"Fatal error: unable to perform malloc");
            exit(7);
         }
         for ( i = 0;  i < lineCnt; i++)
            strList[i] = XmStringCreateLocalized(help2[i]);
         break;
```

---

viewerCallbacks.c++

In function viewerCBdraw:

```
   if (selecting) {
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      GLint viewport[4];
```

```
      glGetIntegerv(GL_VIEWPORT, viewport);
      gluPickMatrix((GLdouble) mousePrevX,
                    (GLdouble) (viewport[3] - mousePrevY),
                    5.0, 5.0, viewport);
      gluPerspective(45.0, aspect,
                     (GLdouble)vattr->viewDistance*0.0005,
                     (GLdouble)vattr->viewDistance*5.0);
   } else if (vattr->viewerShowBoundingBox == YesV) {
      drawBoundingBox(vattr->min, vattr->max,
                      vattr->viewDistance, (int)selecting, mousePrevX,
                      mousePrevY, aspect, vattr->viewDistance);

      (void)sprintf(str1,"E");
      vcDrawText3(1.0, 0.0, 0.0,
                  vattr->max[0], vattr->min[1], vattr->min[2], str1);

      (void)sprintf(str1,"N");
      vcDrawText3(0.0, 1.0, 0.0,
                  vattr->min[0], vattr->max[1], vattr->min[2], str1);

      (void)sprintf(str1,"Up");
      vcDrawText3(0.0, 0.0, 1.0,
                  vattr->min[0], vattr->min[1], vattr->max[2], str1);
   } // endif !selecting
```

---

plotClass_display.c++

Cleaned up representation of tensile strength:

```
in void   PlotClass::display (void):

   else if (attributes.displayMode == Fluid){
               if (sigma1 == sigma3) {
                  dotData[i] = fluidCoef = 0.0;
                  dotDataMax = 1.0;
                  dotDataMin = fluidCoef;
               }
               else {
                  if (normalStress == 0.0)
                     fluidCoef = 0.0;
                  else
                     fluidCoef = attributes.fluidPressure
                            / (normalStress -  attributes.tensileStrength);

                  dotData[i] = fluidCoef;

                  if (fluidCoef > dotDataMax)
                     dotDataMax = fluidCoef;
                  if (fluidCoef < dotDataMin)
                     dotDataMin = fluidCoef;
               }
            }

      else if (attributes.displayMode == Fluid) {
         drawText(attributes.textred,
                  attributes.textgrn,
                  attributes.textblu,
                   70, 125,
                  "FluidP/(NStress-TensileStr)");
      }
```

cmdClass.c++

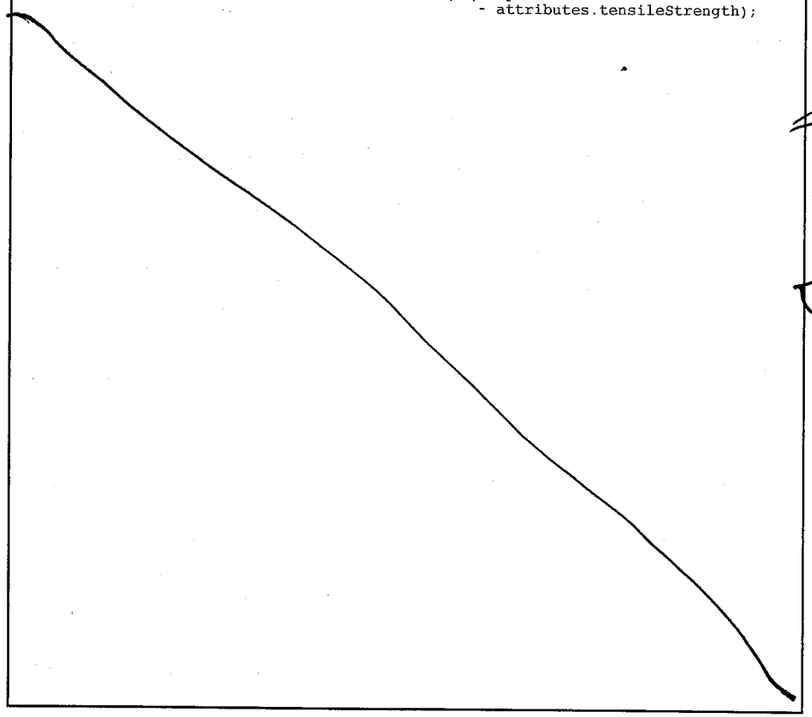
```
void CmdClass::setFluidPress(float value){
   PlotAttrType           plotAttr=plotObj.getAttributes();

   if (value < 0) // lower bound for fluid press
         value = 0;
   if (value > 100) // upper bound for fluid press
```



```
               value = 100;

   XmScaleSetValue(fluidPressW, value * 1000);
   plotAttr.fluidPressure = value;
   plotObj.setAttributes(plotAttr);
   sceneObj.update();

}

void CmdClass::setTensileStr(float value){
   PlotAttrType           plotAttr=plotObj.getAttributes();

   if (value < -40) // lower bound for ten. str.
      value = -40;
   if(value > 0) // upper bound for ten. str.
      value = 0;

   XmScaleSetValue(tensileStrW, value * 1000);
   plotAttr.tensileStrength = value;
   plotObj.setAttributes(plotAttr);
   sceneObj.update();
}
```

plotClass_findSlipV.c++

```
   fluidPressTend = attributes.fluidPressure / (slipNormalStress
                                                - attributes.tensileStrength);
```



plotClass_display.c++

Fixed the azimuth notation by modifiying lines to appear as follows:

```
          if ((float)tmpAz < 10)
             sprintf(strData, "Slip Az       00%3.1f", (float)tmpAz);
          else if ((float)tmpAz < 100)
             sprintf(strData, "Slip Az       0%4.1f", (float)tmpAz);
          else
             sprintf(strData, "Slip Az       %4.1f", (float)tmpAz);
```

---

mohrCallbacks.c++

Changed graphics display "Tensile Strength" to "Computed Tensile Strength"

```
sprintf(str1,"Computed Tensile Strength: %lf", mohrObj.tensileStr);
```

---

mapClass.c++

Added the following line to the very end of the initialize fuction to force drawing of the legend.

```
display();
```


Began modifications to allow using a display list to draw the bounding box in the surface and 3d viewer windows.

boundBox.hh

Added the following lines to facilitate using a display list to draw the bounding box.

```
#define BOUNDING_BOX_LIST   1
// ^ display list for bounding box
```

boundBox.c++

Modified the drawBoundingBox function by removing most of the variable declarations and initializations from the middle of the calculations and put them at the beginning. Also added display list features. This did not work becaue of the d index in some of the later loops.

New Tactic:

boundBox.c++

Simply added the following lines to the beginning and end of the else clause in the top if statement.

```
      glNewList(BOUNDING_BOX_LIST, GL_COMPILE);
      glEndList();  // above lies the definition of BOUNDING_BOX
```

Added the following to improve performance.

```
   if (selecting) {
   } else {
      glNewList(BOUNDING_BOX_LIST, GL_COMPILE);
         glColor3f(0.5, 0.5, 0.5);
         glPolygonMode(GL_BACK, GL_LINE);
         // glPolygonMode(GL_FRONT, GL_POINT);
         glCullFace(GL_FRONT);
         glEnable(GL_CULL_FACE);

      .
      .
      .

      glDisable(GL_CULL_FACE);
      glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
```

Also, removed the matrix circ and substituted glVertex3f for glVertex3fv. Removed all extraneous inputs, and added an id # for the display list as an input.

surfClass.hh

Added the variable SurfYesNoType firstDraw to the attributes to tell whether or not to draw the bounding box.

surfClass.c++

Added initialization of firstDraw to YesS to the constructor.

surfCallbacks.c++

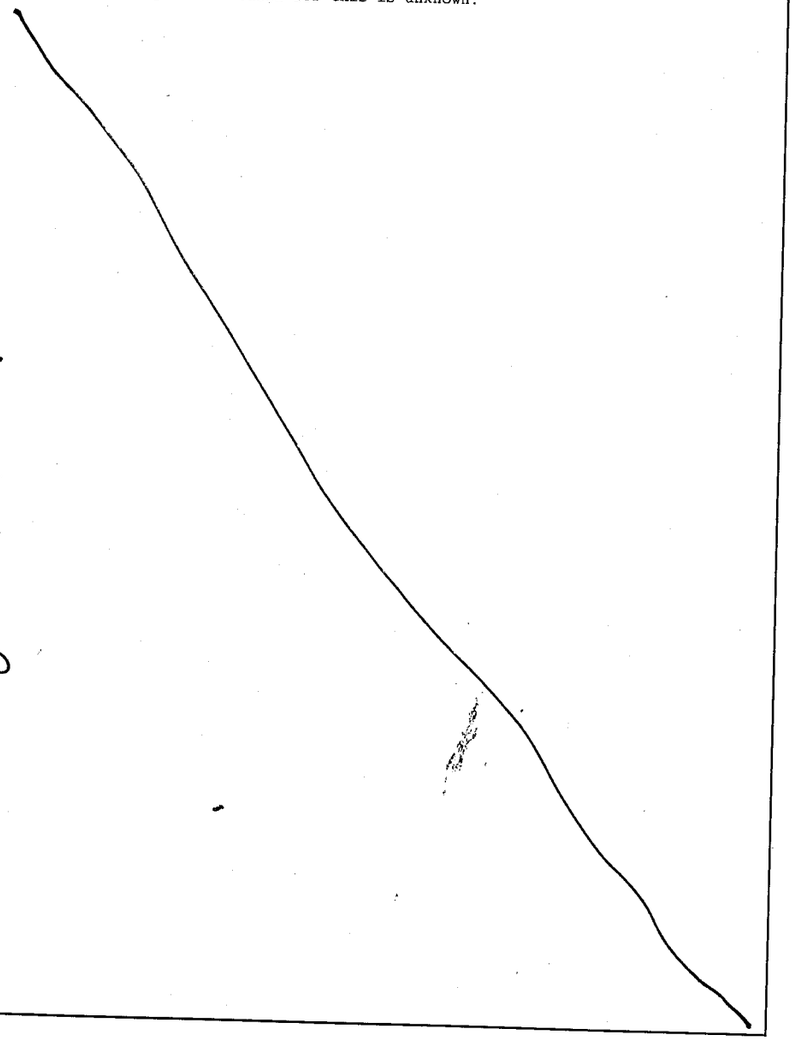
In the function surfCBdraw, added the following.

```
   if (vattr->BoundingBox == YesS) {
      if (vattr->firstDraw == YesS) {
         drawBoundingBox(CIRCLE_LIST, vattr->min, vattr->max);
         vattr->firstDraw = NoS;
      }
```

Printed by jbuckner from yosemite


```
    glCallList(CIRCLE_LIST);
  } // end if BoundingBox==YesS
```

Determined that the display list compilation was causing significant errors in the drawing of the legend in both the 3dviewer and the surface windows. The specific reason for this is unknown.



Scrapped all of the changes made to facillitate the use of display lists for bounding boxes. Robert Boenau made a new bounding box class.

surfClass.c++

Implemented changes to use the new bounding box class.

```
#include "boundBoxObjs.hh"
```

In the function void SurfClass::initialize(Widget inWidget) added the following lines.

```
    // set the size of the bounding box
    surfBox.setMinMax(attributes.min,  attributes.max);
```

surfCallbacks.c++

Removed unnecessary line:

```
static    int        selecting = 0;
```

```
#include "boundBoxObjs.hh"
```

Implemented changes to use the new bounding box class.

In the function void surfCBdraw(GLfloat rotMat[16]) modified the lines dealing with the bounding box to read as follows.

```
  if (vattr->BoundingBox == YesS) {  // shall we draw the bounding box?
    surfBox.display();  // draw the bounding box

    (void)sprintf(str1,"E");
    scDrawText3(1.0, 0.0, 0.0, 115, 0.0, 0.0, str1);

    (void)sprintf(str1,"W");
    scDrawText3(1.0, 0.0, 0.0, -115, 0.0, 0.0, str1);

    (void)sprintf(str1,"N");
    scDrawText3(0.0, 1.0, 0.0, 0.0, 115, 0.0, str1);

    (void)sprintf(str1,"S");
    scDrawText3(0.0, 1.0, 0.0, 0.0, -115, 0.0, str1);
  } // end if BoundingBox==YesS
```

viewerClass.c++

Implemented changes to use the new bounding box class.

```
#include "boundBoxObjs.hh"
```

In the function void ViewerClass::initialize(Widget inWidget) added the following lines.

```
    // set the size of the bounding box
    viewerBox.setMinMax(attributes.min,  attributes.max);
```

viewerCallbacks.c++

```
#include "boundBoxObjs.hh"
```

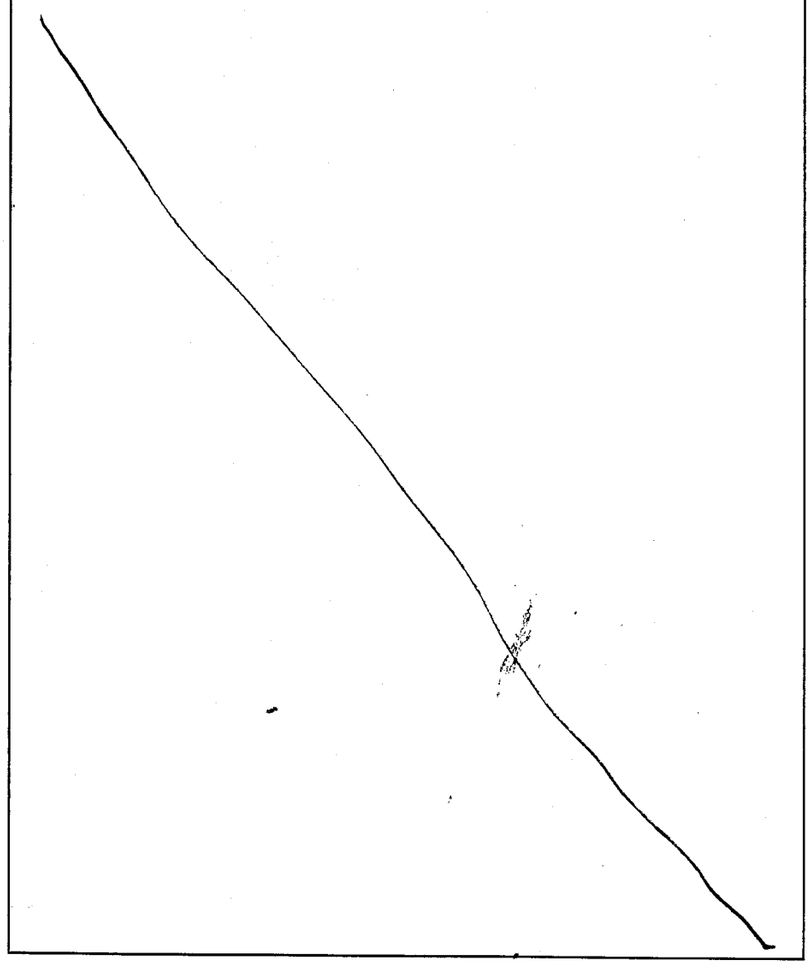
Implemented changes to use the new bounding box class.

In the function void viewerCBdraw(GLfloat rotMat[16]) modified the lines dealing with the bounding box to read as follows.

```
  } else if (vattr->viewerShowBoundingBox == YesV) {
    viewerBox.display();  // draw the bounding box

    (void)sprintf(str1,"E");
    vcDrawText3(1.0, 0.0, 0.0,
                vattr->max[0], vattr->min[1], vattr->min[2], str1);
```

JWB

```
    (void)sprintf(str1,"N");
    vcDrawText3(0.0, 1.0, 0.0,
                vattr->min[0], vattr->max[1], vattr->min[2], str1);

    (void)sprintf(str1,"Up");
    vcDrawText3(0.0, 0.0, 1.0,
                vattr->min[0], vattr->min[1], vattr->max[2], str1);
} // endif !selecting
```



JWB

mapCallbacks.c++

Removed the line #include "boundBox.hh" since no function from that particular header appears in the code.

Makefile

Added the boundBoxClass by simply replacing all boundBox occurences with boundBoxClass. Then discoved that the necessary changes had already been made by rboenau. So, scrapped changes.

viewerButtonCB.c++

```
#include "boundBoxObjs.hh"
```

In the function openFltFile, added the following line.

```
      viewerBox.setMinMax(vattr->min,  vattr->max);
```

viewerOptionCB.c++

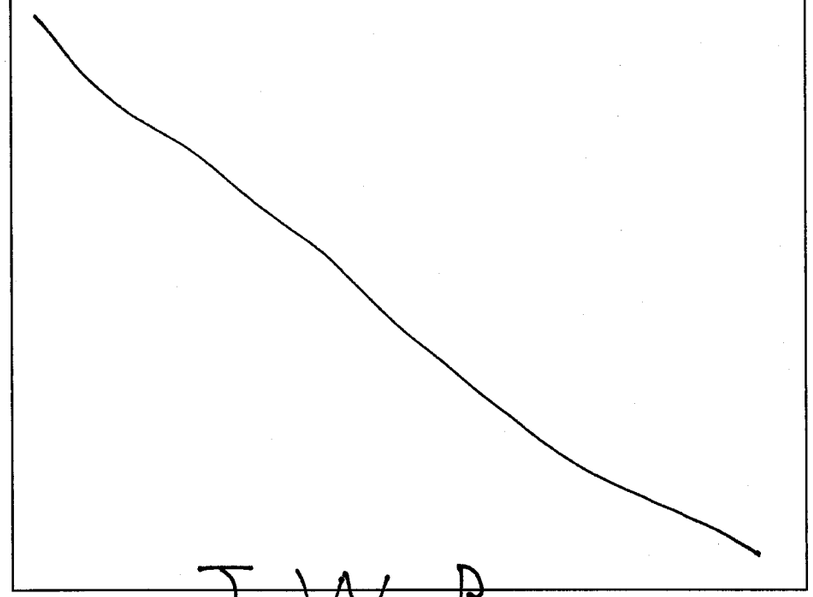
```
#include "boundBoxObjs.hh"
```

In the function viewerRemoveCB, added the following line

```
      viewerBox.setMinMax(vattr->min,  vattr->max);
```

viewNetGlobals.hh

```
#include "boundBoxClass.hh"

boundBoxClass viewerBox;
boundBoxClass surfBox;
```



JWB

viewerButtonCB.c++

In function void viewerHelpPress, removed the name of the second argument previously labeled XtPointer clientdata to avoid the following compiler warning.

```
"viewerButtonCB.c++", line 449: warning(3262): parameter "clientData" declared
          and never referenced
  void viewerHelpPress(Widget, XtPointer clientData, XtPointer){
                               ^
```

viewerOptionClass.hh

Added widget graphStepSize to the attributes structure for a step size scale on the bounding box grid.

```
Widget   gridStepSize;
```

viewerOptionClass.c++

Added widget graphStepSize to the initialization function.

```
attributes.gridStepSize = XtVaCreateManagedWidget("gridStepSize",
                                    xmScaleWidgetClass,
                                    col9,
                                    XtVaTypedArg,
                                    XmNtitleString,
                                    XmRString, "Grid Step Size", 15,
                                    XmNorientation, XmHORIZONTAL,
                                    XmNmaximum, 100,
                                    XmNminimum, 2,
                                    XmNscaleMultiple, 1,
                                    XmNvalue, 10,
                                    XmNscaleHeight, 20,
                                    XmNscaleWidth, 100,
                                    XmNshowValue, True,
                                    XmNbackground, WIDGET_COLOR,
                                    NULL);

XtAddCallback(attributes.gridStepSize, XmNvalueChangedCallback,
              changeGridStepSize, NULL);
XtAddCallback(attributes.gridStepSize, XmNdragCallback,
              changeGridStepSize, NULL);
```

viewerOptionCB.hh

Added a callback for the new gridStepSize scale widget.

```
void changeGridStepSize(Widget, XtPointer, XtPointer);
```

viewerOptionCB.c++

Added a callback for the new gridStepSize scale widget.

```
void changeGridStepSize(Widget, XtPointer, XtPointer call_data){

   XmScaleCallbackStruct *value = (XmScaleCallbackStruct *) call_data;

   viewerBox.setStepSize((float)(value->value));
   viewerCBdraw(noRot);
} // endof changeViewerAxisLength
```


surfClass.c++

Removed references to the bounding box class including the following line.

```
#include "boundBoxObjs.hh"
```

surfCallbacks.c++

Revised the function surfCBdraw to resize the bounding box when the verticle exageration changes. The code now appears as follows.

```
   if (vattr->BoundingBox == YesS) {  // shall we draw the bounding box?
      float newMax[3];

      newMax[0] = Xdata[0];
      newMax[1] = Ydata[0];
      newMax[2] = Zdata[0] * vattr->vert;

      for (index = 1; index < 36000; index++) {

         if (Xdata[index] > newMax[0])
            newMax[0] = Xdata[index];

         if (Ydata[index] > newMax[1])
            newMax[1] = Ydata[index];

         if ( (Zdata[index] * vattr->vert) > newMax[2])
            newMax[2] = Zdata[index] * vattr->vert;
      }

      if (newMax[2] <= 0.0)
         newMax[2] = 1.0;

      surfBox.setMinMax(vattr->min,  newMax);
      surfBox.display();  // draw the bounding box
```

surfOptionCB.c++

Removed references to the bounding box class including the following line.

```
#include "boundBoxObjs.hh"
```

viewNet.c++

Changed the "what" strings to the following

```
char ident[] = "@(#)3dstress 10-30-96 Rev 1.3 alpha";
char develop[] = "@(#)Code written by: Brent Henderson,  Robert Boenau,  and Joshu
a Buckner";
```

==========

Toyed with extending the normal vector through to the other side of the triangle in the 3d viewer. Discovered that this requires the use of the following two equation for use in 3d geometry.

```
(x1 * y1 + z1 * y2) * x + (z1 * x2 - x1 * z2) * y
   + (x1 * y2 - y1 * x2) * z = 0

(x2 - x) * (x2 - x) + (y2 - y) * (y2 - y) + (z2 - z) * (z2 - z)
   = (x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) + (y3 - y1) * (y3 - y1)
```


pointClass.hh

Added another set of coordinates for a point in 3-space.

```
double otherEnd[3];
```

segmentClass.c++

In the function recalculateVectors(), added the following lines to allow the normal vector to be double sided.

```
VectorClass otherPoint;

        otherPoint = normalVector.scale(-vectorLengths);
        otherPoint = otherPoint.add(center);
        otherPoint.get(&(tmpArray[0]),
                       &(tmpArray[1]),
                       &(tmpArray[2]) );

        points[i].otherEnd[0] = (double)tmpArray[0];
        points[i].otherEnd[1] = (double)tmpArray[1];
        points[i].otherEnd[2] = (double)tmpArray[2];
```

In the function calculateVector(), added the following lines to allow the normal vector to be double sided.

```
VectorClass otherPoint;

        otherPoint = normalVector.scale(-vectorLengths);
        otherPoint = otherPoint.add(center);
        otherPoint.get(&(tmpArray[0]),
                       &(tmpArray[1]),
                       &(tmpArray[2]) );

        points[hitname].otherEnd[0] = (double)tmpArray[0];
        points[hitname].otherEnd[1] = (double)tmpArray[1];
        points[hitname].otherEnd[2] = (double)tmpArray[2];
```

In the function display, add the following lines where normal vectors are draw.

```
                glBegin(GL_LINES);
                   glVertex3dv(points[i].otherEnd);
                   glVertex3dv(points[i].center);
                glEnd();
```

==========

viewerClass.hh

Added variable to keep track of double sided norm toggle status.

```
int        doubleNorm;
```

viewerClass.c++

Added initialization of the doubleNorm attribute variable.

```
attributes.doubleNorm = 1;
```

viewerOptionClass.hh

Added widget for the doubleNorm toggle.

```
Widget vectorDoubleToggle;
```

viewerOptionClass.c++

```
attributes.vectorDoubleToggle = XtVaCreateManagedWidget("VDouble",
                                    xmToggleButtonGadgetClass,
                                    col5,
```

22Aug

```
                                    XtVaTypedArg,
                                    XmNlabelString,
                                    XmRString, "Double Sided Vector", 20,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

if (vattr->doubleVector == 1)
   XmToggleButtonSetState(attributes.vectorDoubleToggle, TRUE, TRUE);


XtAddCallback (attributes.vectorDoubleToggle, XmNvalueChangedCallback,
               viewervectorDoubleToggleChange, (XtPointer)0);
```


viewerOptionClass.c++

Corrected a line dealing with the new double sided norm toggle.

```
if (vattr->doubleNorm == 1)
   XmToggleButtonSetState(attributes.vectorDoubleToggle, TRUE, TRUE);
```

viewerOptionCB.hh

Added a callback to deal with the new double sided norm toggle.

```
void viewerVectorDoubleToggleChange(Widget, XtPointer, XtPointer);
```

viewerOptionCB.c++

Added a callback to deal with the new double sided norm toggle.

```
void viewerVectorDoubleToggleChange(Widget, XtPointer, XtPointer){
   ViewerAttributeType *vattr = viewerObj.getAttributes();
   if (vattr->doubleNorm == 1)
      vattr->doubleNorm = 0;
   else
      vattr->doubleNorm = 1;

   viewerCBdraw(noRot);
} // endof changeViewerAxis
```

segmentClass.c++

Added an include to allow testing of the doubleNorm variable in the viewerClass attributes within the void SegmentClass::display function.

```
#include "viewerClass.hh"
#include "viewerObj.hh"

void SegmentClass::display(RenderType renderMode, int format, int recalc,
                           PlotColorTableType *ct, float cscale,
                           vectorStruct *vectorInfo){
.
.
.

ViewerAttributeType * vattr = viewerObj.getAttributes();

.
.
.

      if (vectorInfo->showNorm) {
                  glColor3fv(vectorInfo->nrgb);
                  glBegin(GL_LINES);
                     glVertex3dv(points[i].center);
                     glVertex3dv(points[i].endLine);
                  glEnd();

                  if(vattr->doubleNorm) {
                     glBegin(GL_LINES);
                        glVertex3dv(points[i].otherEnd);
                        glVertex3dv(points[i].center);
                     glEnd();
                  }

.
.
.
```

---

pointClass.hh

Added another variable to take care of displaying the opposite of the

```
current slipVect (for Chevron's double sided vectors).

      double oppositeSlipVect[3];

segmentClass.c++

   In the functions calculateVector and recalculateVectors, added
calculation of the opposite slip vector (as per Chevron's wishes)

        // calculate the other side of the slip vector
         slipVector.set(tmpSlipVect[0], tmpSlipVect[1], tmpSlipVect[2]);
         slipVector.normal();
         slipVector = slipVector.scale(-vectorLengths);
         slipVector = slipVector.add(center);

         // Store data into the structure
         slipVector.get(&(tmpArray[0]),
                        &(tmpArray[1]),
                        &(tmpArray[2]) );

         points[?].oppositeSlipVect[0] = (double)tmpArray[0];
         points[?].oppositeSlipVect[1] = (double)tmpArray[1];
         points[?].oppositeSlipVect[2] = (double)tmpArray[2];

   In the function
void SegmentClass::display(RenderType renderMode, int format, int recalc,
                           PlotColorTableType *ct, float cscale,
                           vectorStruct *vectorInfo),
added drawing of opposite side of slipVector.

              if (vectorInfo->showNorm) {
                 glColor3fv(vectorInfo->nrgb);
                 glBegin(GL_LINES);
                    glVertex3dv(points[i].center);
                    glVertex3dv(points[i].endLine);
                 glEnd();

                 if (vattr->doubleNorm) {
                    glBegin(GL_LINES);
                       glVertex3dv(points[i].oppositeSlipVector);
                       glVertex3dv(points[i].center);
                    glEnd();
                 }
              }

viewerClass.hh

   Added toggle for displaying of double sided slip vector to the
attributes structure.

   int doubleSlip;

viewerClass.c++

   Added initialization of doubleSlip vector.

   attributes.doubleSlip = 1;

viewerOptionClass.hh

   Added widget toggle for controlling the doubleSlip toggle.

Widget slipDoubleToggle

viewerOptionClass.c++

   Added widget toggle for controlling the doubleSlip toggle and callbacks.

   attributes.slipDoubleToggle = XtVaCreateManagedWidget("SDouble",
                                    xmToggleButtonGadgetClass,
                                    col5,
```



```
                                    XtVaTypedArg,
                                    XmNlabelString,
                                    XmRString, "Double Sided Slip", 18,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

   if (vattr->doubleNorm == 1)
      XmToggleButtonSetState(attributes.slipDoubleToggle, TRUE, TRUE);

   XtAddCallback (attributes.slipDoubleToggle, XmNvalueChangedCallback,
                  viewerSlipDoubleToggleChange, (XtPointer)0);

viewerOptionCB.hh

   Added callback for the slipDoubleToggle widget.

void viewerSlipDoubleToggleChange(Widget, XtPointer, XtPointer);

viewerOptionCB.c++

   Added callback for the slipDoubleToggle widget.

void viewerSlipDoubleToggleChange(Widget, XtPointer, XtPointer){
   ViewerAttributeType *vattr = viewerObj.getAttributes();
   if (vattr->doubleSlip == 1)
      vattr->doubleSlip = 0;
   else
      vattr->doubleSlip = 1;

   viewerCBdraw(noRot);
} // endof viewerSlipDoubleToggleChange
```

controlClass.hh

Added integer to keep track of the orientation (vertical/horizontal) of the controll bar to the attributes structure.

```
int                     orientation;  // 0 = horiz., 1 = vert.
```

optionClass.hh

Added widget for toggling the orientation (vertical/horizontal) of the controll bar to the attributes structure.

```
Widget orientationButton;
```

optionClass.c++

Added, to the initialize function, initialization of a widget and its callback for toggling the orientation (vertical/horizontal) of the controll bar.

```
attributes.orientationButton = XtVaCreateManagedWidget("OButton",
                                    xmToggleButtonGadgetClass,
                                    colWidget,
                                    XmNlabelString, orientStr,
                                    XmNselectColor, BUTTON_COLOR,
                                    NULL);

XtAddCallback(attributes.orientationButton, XmNvalueChangedCallback,
              optionOrientationChangeCB, (XtPointer)0);
```

optionCallbacks.hh

Added the callback for toggling the orientation (vertical/horizontal) of the controll bar.

```
void optionOrientationChangeCB(Widget, XtPointer, XtPointer);
```

optionCallbacks.c++

Added a shell for the callback for toggling the orientation (vertical/horizontal) of the controll bar.

```
void optionOrientationChangeCB(Widget, XtPointer, XtPointer) {
}
```


Scrapped toggle for changing the orientation the controll bar since the vertical controll bar wouldn't work correctly and the integrity of the code suffered from the some of the necessary changes.

==========

Discovered a problem with the 3dviewer bounding box. It dumped the core when the 'b' key was pressed.

~3dstress/3d/help/MohrGraph.sc

Added to the help file on the topic of the 3D Fault Viewer.

J W B

file modified: cmdClass_toggleCB.c++

function modified: void loadMag(Widget w, XtPointer, XtPointer call_data)

modifications made:

```
if (values[9] < 1 || values[9] > 3) changed to
if (values[9] < 0 || values[9] > 3)
```

Purpose for modifications:

Previous to modification, the user was unable to load a file of sigma magnitudes if that file was saved before an initial choice of sigma direction and plunge was made. Now, the user may load a file without a chosen direction and plunge.

J W B

File: mohrOptionCB.c++

Function: void mohrChanges3Tos1RatioField(Widget, XtPointer, XtPointer)

Original Lines:

```
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 1000);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 1000);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 1000);
```

After Modifications:

```
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 100);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 100);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 100);
```

Function: void mohrChanges2Tos1RatioField(Widget, XtPointer, XtPointer)

Original Lines:

```
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 1000);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 1000);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 1000);
```

After Modifications:

```
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 100);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 100);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 100);
```

Function:

```
   void mohrs3Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
```

Original Lines:

```
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 1000);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 1000);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 1000);
```

After Modification:

```
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 100);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 100);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 100);
```

Function:

```
   void mohrs2Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
```

Original Lines:

```
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 1000);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 1000);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 1000);
```

After Modification:

```
   XmScaleSetValue(moattr->scale[max], mattr->sigmas[max] * 100);
   XmScaleSetValue(moattr->scale[mid], mattr->sigmas[mid] * 100);
   XmScaleSetValue(moattr->scale[min], mattr->sigmas[min] * 100);
```

Function:

```
void mohrChangeMinMax(Widget, XtPointer clientData, XtPointer)
```

Original Lines:

```
   for (int i = 0; i < 3; i++) {
      if (mattr->sigmas[i] < mattr->minScale ||
          mattr->sigmas[i] > mattr->maxScale) {
```

J W B


```
         mattr->sigmas[i] = mattr->minScale;
         XmScaleSetValue(moattr->scale[i], mattr->sigmas[i] * 1000);
      } // end if
      XtVaSetValues(moattr->scale[i],
                    XmNmaximum, mattr->maxScale * 1000,
                    XmNminimum, mattr->minScale * 1000,
                    NULL);
   } // end for int i
```

After Modification:

```
   for (int i = 0; i < 3; i++) {
      if (mattr->sigmas[i] < mattr->minScale ||
          mattr->sigmas[i] > mattr->maxScale) {
         mattr->sigmas[i] = mattr->minScale;
         XmScaleSetValue(moattr->scale[i], mattr->sigmas[i] * 100);
      } // end if
      XtVaSetValues(moattr->scale[i],
                    XmNmaximum, mattr->maxScale * 100,
                    XmNminimum, mattr->minScale * 100,
                    NULL);
   } // end for int i
```

Function:

```
   void mohrToggleStressMode(Widget, XtPointer clientData, XtPointer)
```

Original Lines:

```
      XmScaleSetValue(optionAttrib->scale[0], mohrAttrib->sigmas[0] * 1000);
      XmScaleSetValue(optionAttrib->scale[1], mohrAttrib->sigmas[1] * 1000);
      XmScaleSetValue(optionAttrib->scale[2], mohrAttrib->sigmas[2] * 1000);
```

After Modification:

```
      XmScaleSetValue(optionAttrib->scale[0], mohrAttrib->sigmas[0] * 100);
      XmScaleSetValue(optionAttrib->scale[1], mohrAttrib->sigmas[1] * 100);
      XmScaleSetValue(optionAttrib->scale[2], mohrAttrib->sigmas[2] * 100);
```

Function:

```
   void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
```

Original Lines:

```
      XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max] *
                                                 1000);
      XmScaleSetValue(optionsAttrib->scale[mid], mohrObjAttrib->sigmas[mid] *
                                                 1000);
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min] *
                                                 1000);
```

After Modification:

```
      XmScaleSetValue(optionsAttrib->scale[max], mohrObjAttrib->sigmas[max] *
                                                 100);
      XmScaleSetValue(optionsAttrib->scale[mid], mohrObjAttrib->sigmas[mid] *
                                                 100);
      XmScaleSetValue(optionsAttrib->scale[min], mohrObjAttrib->sigmas[min] *
                                                 100);
```

File: mohrOptionClass.c++

Function:

```
   void MohrOptionClass::initialize(Widget w)
```

Original Lines:

```
   XmString s1;
```


J W B

```
    s1 = motifFontObj.create("", "s", " U Magnitude");
    attributes.scale[0] = XtVaCreateManagedWidget("U",
                                xmScaleWidgetClass,
                                col2,
                                XmNtitleString, s1,
                                XmNfontList, motifFontObj.fontListing,
                                XmNorientation, XmHORIZONTAL,
                                XmNmaximum, 100000,
                                XmNminimum, -100000,
                                XmNdecimalPoints, 3,
                                XmNvalue, 0,
                                XmNscaleMultiple, 1,
                                XmNshowValue, True,
                                XmNbackground, WIDGET_COLOR,
                                NULL);

    XmStringFree(s1);
    s1 = motifFontObj.create("", "s", " V Magnitude");

    attributes.scale[1] = XtVaCreateManagedWidget("V",
                                xmScaleWidgetClass,
                                col2,
                                XmNtitleString, s1,
                                XmNfontList, motifFontObj.fontListing,
                                XmNorientation, XmHORIZONTAL,
                                XmNmaximum, 100000,
                                XmNminimum, -100000,
                                XmNdecimalPoints, 3,
                                XmNvalue, 50000,
                                XmNscaleMultiple, 1,
                                XmNshowValue, True,
                                XmNbackground, WIDGET_COLOR,
                                NULL);

    XmStringFree(s1);
    s1 = motifFontObj.create("", "s", " W Magnitude");
    attributes.scale[2] = XtVaCreateManagedWidget("W",
                                xmScaleWidgetClass,
                                col2,
                                XmNtitleString, s1,
                                XmNfontList, motifFontObj.fontListing,
                                XmNorientation, XmHORIZONTAL,
                                XmNmaximum, 100000,
                                XmNminimum, -100000,
                                XmNdecimalPoints, 3,
                                XmNvalue, 100000,
                                XmNscaleMultiple, 1,
                                XmNshowValue, True,
                                XmNbackground, WIDGET_COLOR,
                                NULL);

    XmStringFree(s1);
```

After Modification:

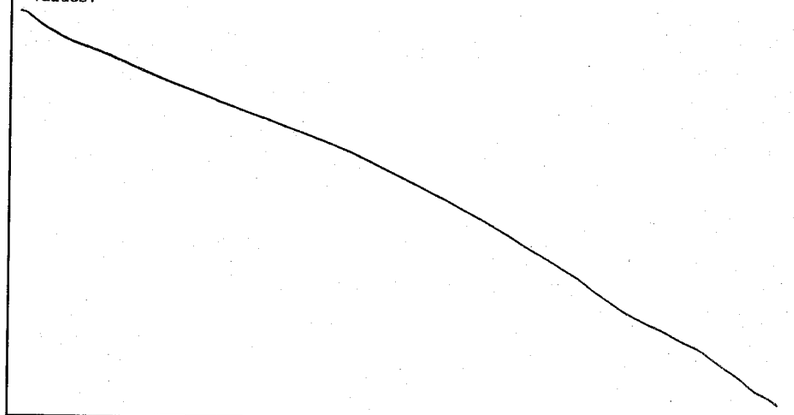
```
    XmString s1;

    s1 = motifFontObj.create("", "s", " U Magnitude");
    attributes.scale[0] = XtVaCreateManagedWidget("U",
                                xmScaleWidgetClass,
                                col2,
                                XmNtitleString, s1,
                                XmNfontList, motifFontObj.fontListing,
                                XmNorientation, XmHORIZONTAL,
                                XmNmaximum, 10000,
                                XmNminimum, -10000,
                                XmNdecimalPoints, 2,
                                XmNvalue, 0,
                                XmNscaleMultiple, 1,
                                XmNshowValue, True,
                                XmNbackground, WIDGET_COLOR,
```

J W B


```
                                NULL);

    XmStringFree(s1);
    s1 = motifFontObj.create("", "s", " V Magnitude");

    attributes.scale[1] = XtVaCreateManagedWidget("V",
                                xmScaleWidgetClass,
                                col2,
                                XmNtitleString, s1,
                                XmNfontList, motifFontObj.fontListing,
                                XmNorientation, XmHORIZONTAL,
                                XmNmaximum, 10000,
                                XmNminimum, -10000,
                                XmNdecimalPoints, 2,
                                XmNvalue, 5000,
                                XmNscaleMultiple, 1,
                                XmNshowValue, True,
                                XmNbackground, WIDGET_COLOR,
                                NULL);

    XmStringFree(s1);
    s1 = motifFontObj.create("", "s", " W Magnitude");
    attributes.scale[2] = XtVaCreateManagedWidget("W",
                                xmScaleWidgetClass,
                                col2,
                                XmNtitleString, s1,
                                XmNfontList, motifFontObj.fontListing,
                                XmNorientation, XmHORIZONTAL,
                                XmNmaximum, 10000,
                                XmNminimum, -10000,
                                XmNdecimalPoints, 2,
                                XmNvalue, 10000,
                                XmNscaleMultiple, 1,
                                XmNshowValue, True,
                                XmNbackground, WIDGET_COLOR,
                                NULL);

    XmStringFree(s1);
```

Purpose for Modifications:

The units of the numbers displayed at the top of the Mohr Graph and in the options window were truncated as per Dr. Ferrill's instructions. For example, fluid pressure is shown to two decimal places as are the sigma values.



J W B

File: mohrOptionClass.c++

Function: void MohrOptionClass::initialize(Widget w)

Original Lines:

```
   attributes.s3Tos1RatioScale = XtVaCreateManagedWidget("ratio1Scale",
                              xmScaleWidgetClass, attributes.colStress2,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, mattr->stressScaleMax,
                              XmNminimum, mattr->stressScaleMin,
                              XmNvalue, (int) (mattr->s3Tos1Ratio * 100),
                              XmNscaleMultiple, 1,
                              XmNdecimalPoints, 2,
                              XmNshowValue, False,

   attributes.s2Tos1RatioScale = XtVaCreateManagedWidget("ratio2Scale",
                              xmScaleWidgetClass, attributes.colStress2,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, mattr->stressScaleMax,
                              XmNminimum, mattr->stressScaleMin,
                              XmNvalue, (int) (mattr->s2Tos1Ratio * 100),
                              XmNscaleMultiple, 1,
                              XmNdecimalPoints, 2,
                              XmNshowValue, False,
                              XmNbackground, WIDGET_COLOR,  NULL);
         XmNbackground, WIDGET_COLOR,  NULL);

   sprintf(strHolder, "%lf", mattr->s3Tos1Ratio);

   sprintf(strHolder, "%lf", mattr->s2Tos1Ratio);

   sprintf(strHolder, "%d", mohrObj.fluidPressure);

   attributes.fluidScale = XtVaCreateManagedWidget("fluidScale",
                              xmScaleWidgetClass, col3,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, mattr->maxScale - 1,
                              XmNminimum, 0,
                              XmNvalue, (int) mohrObj.fluidPressure,
                              XmNscaleMultiple, 1,
                              XmNshowValue, False,
                              XmNbackground, WIDGET_COLOR,  NULL);
```

After Modification:

```
   attributes.s3Tos1RatioScale = XtVaCreateManagedWidget("ratio1Scale",
                              xmScaleWidgetClass, attributes.colStress2,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, mattr->stressScaleMax * 10,
                              XmNminimum, mattr->stressScaleMin * 10,
                              XmNvalue, (int) (mattr->s3Tos1Ratio * 1000),
                              XmNscaleMultiple, 1,
                              XmNdecimalPoints, 3,
                              XmNshowValue, False,
                              XmNbackground, WIDGET_COLOR,  NULL);


   attributes.s2Tos1RatioScale = XtVaCreateManagedWidget("ratio2Scale",
                              xmScaleWidgetClass, attributes.colStress2,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, mattr->stressScaleMax * 10,
                              XmNminimum, mattr->stressScaleMin *10,
                              XmNvalue, (int) (mattr->s2Tos1Ratio * 1000),
                              XmNscaleMultiple, 1,
                              XmNdecimalPoints, 3,
                              XmNshowValue, False,
                              XmNbackground, WIDGET_COLOR,  NULL);


   sprintf(strHolder, "%5.3lf", mattr->s3Tos1Ratio);
```


J W B.


```
   sprintf(strHolder, "%5.3lf", mattr->s2Tos1Ratio);

   sprintf(strHolder, "%.2lf", mohrObj.fluidPressure);

   attributes.fluidScale = XtVaCreateManagedWidget("fluidScale",
                              xmScaleWidgetClass, col3,
                              XmNorientation, XmHORIZONTAL,
                              XmNmaximum, (mattr->maxScale * 100) - 1,
                              XmNminimum, 0,
                              XmNvalue, (int) (mohrObj.fluidPressure * 100.0),
                              XmNscaleMultiple, 1,
                              XmNdecimalPoints, 2,
                              XmNshowValue, False,
                              XmNbackground, WIDGET_COLOR,  NULL);
```

File: mohrOptionCB.c++

Function:

```
   void mohrScaleChange(Widget, XtPointer clientData, XtPointer callData)
```

Original Lines:

```
   double
      newValue = (double) cbs->value / 1000;
```

After Modification:

```
   double
      newValue = (double) cbs->value / 100;
```

Function:

```
   void mohrChanges3Tos1RatioField(Widget, XtPointer, XtPointer)
```

Original Lines:

```
      sprintf(strHolder,"%5lf", mattr->s3Tos1Ratio);

   XmScaleSetValue(moattr->s3Tos1RatioScale, (int) (mattr->s3Tos1Ratio * 100));
```

After Modification:

```
      sprintf(strHolder,"%5.3lf", mattr->s3Tos1Ratio);

   XmScaleSetValue(moattr->s3Tos1RatioScale, (int) (mattr->s3Tos1Ratio * 1000));
```

Function:

```
   void mohrs3Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
```

Original Lines:

```
      sprintf(strHolder,"%5lf", mattr->s3Tos1Ratio);

   mattr->s3Tos1Ratio =  (double) (cbs->value / 100.0);

      XmScaleSetValue(moattr->s3Tos1RatioScale,
                  (int) (mattr->s3Tos1Ratio * 100));
```

After Modification:

```
      sprintf(strHolder,"%5.3lf", mattr->s3Tos1Ratio);

   mattr->s3Tos1Ratio =  (double) (cbs->value / 1000.0);

      XmScaleSetValue(moattr->s3Tos1RatioScale,
                  (int) (mattr->s3Tos1Ratio * 1000));
```


J W B

Function: void mohrChanges2Tos1RatioField(Widget, XtPointer, XtPointer)

Original Lines:

```
        sprintf(strHolder,"%5lf", mattr->s2Tos1Ratio);
   XmScaleSetValue(moattr->s2Tos1RatioScale, (int) (mattr->s2Tos1Ratio * 100));
```

After Modification:

```
        sprintf(strHolder,"%5.3lf", mattr->s2Tos1Ratio);
   XmScaleSetValue(moattr->s2Tos1RatioScale, (int) (mattr->s2Tos1Ratio * 1000));
```

Function:

```
   void mohrs2Tos1RatioScaleChange(Widget, XtPointer, XtPointer callData)
```

Original Lines:

```
      sprintf(strHolder,"%5lf", mattr->s2Tos1Ratio);
   mattr->s2Tos1Ratio =  (double) (cbs->value / 100.0);
      XmScaleSetValue(moattr->s2Tos1RatioScale,
                      (int) (mattr->s3Tos1Ratio * 100));
```

After Modification:

```
      sprintf(strHolder,"%5.3lf", mattr->s2Tos1Ratio);
   mattr->s2Tos1Ratio =  (double) (cbs->value / 1000.0);
      XmScaleSetValue(moattr->s2Tos1RatioScale,
                      (int) (mattr->s3Tos1Ratio * 1000));
```

Function: void updateEffRatios()

Original Line: `sprintf(str, "%5lf", holder);`

After Modification: `sprintf(str, "%5.3lf", holder);`

Function: void updateEffective()

Original Line: `sprintf(strs[i],"%5lf",mattr->effStress[i]);`

After Modification: `sprintf(strs[i],"%5.2lf",mattr->effStress[i]);`

Function: void mohrChangeFluidField(Widget, XtPointer, XtPointer)

Original Line:

```
   sprintf(strHolder,"%5lf", mohrObj.fluidPressure);
   XmScaleSetValue(moattr->fluidScale, mohrObj.fluidPressure);
```

After Modifications:

```
   sprintf(strHolder,"%5.2lf", mohrObj.fluidPressure);
   XmScaleSetValue(moattr->fluidScale, (int) (mohrObj.fluidPressure * 100.0));
```

Function:

```
   void mohrChangeFluidScale(Widget, XtPointer, XtPointer callData)
```

Original Line:

```
   mohrObj.fluidPressure =  (double) (cbs->value);
   sprintf(strHolder,"%5lf", mohrObj.fluidPressure);
```


J W B.

After Modifications:

```
   mohrObj.fluidPressure =  (double) ((cbs->value) / 100.0);
   sprintf(strHolder,"%5.2lf", mohrObj.fluidPressure);
```

File: mohrCallbacks.c++

Function: void mohrCBdraw()

Original Lines:

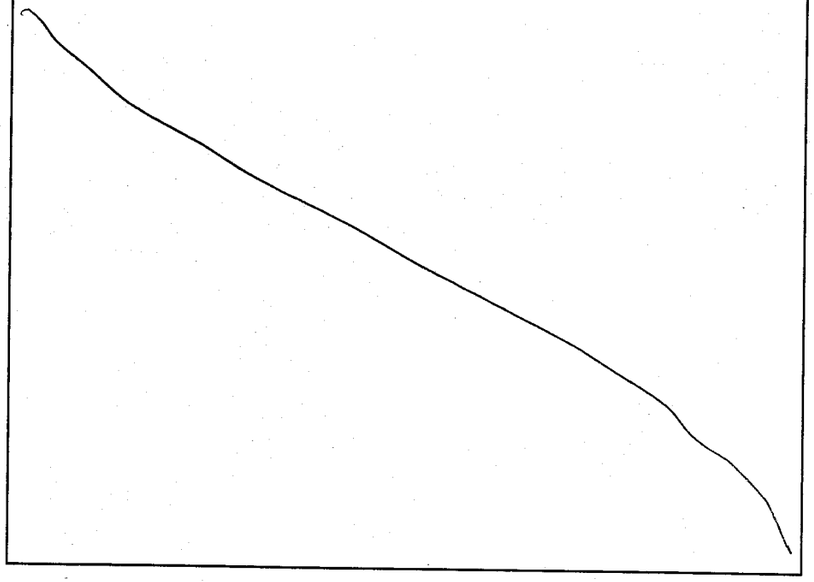
```
   sprintf(str1,"Fluid Pressure: %lf", mohrObj.fluidPressure);
   sprintf(str1,"Computed Tensile Strength: %lf", mohrObj.tensileStr);
   motifFontObj.drawGLFont(0.0, 1.0, 1.0, 0.5, 18.0,
                           "Tensile Strength: undefined");
```

After Modification:

```
   sprintf(str1,"Fluid Pressure: %.2lf", mohrObj.fluidPressure);
   sprintf(str1,"Comp Tensile Str: %.2lf", mohrObj.tensileStr);
   motifFontObj.drawGLFont(0.0, 1.0, 1.0, 0.5, 18.0,
                           "Tensile Str: undefined");
```

Purpose for Modifications:

The units of the numbers displayed at the top of the Mohr Graph and in the options window were truncated as per Dr. Ferrill's instructions. For example, fluid pressure is shown to two decimal places as are the sigma values.



J W B

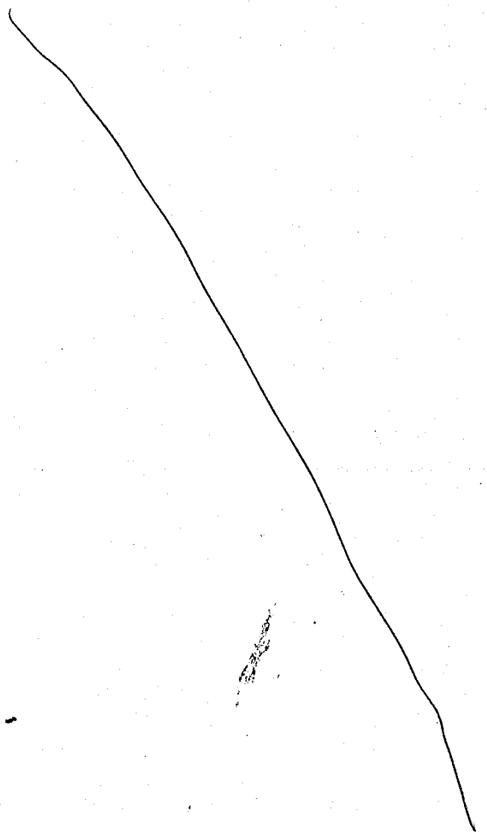
File: plotClass.c++

Function: void PlotClass::calcColorsFluid (double rad,
GLfloat *red,
GLfloat *grn,
GLfloat *blu)

Original Lines:

```
// Light orange to light green
switch ((int)i) {
   case 0:
      *red = 0.2;
      *grn = 1.0;
      *blu = 0.2;
      break;
   case 1:
      *red = 1.0;
      *grn = 1.0;
      *blu = 0.4;
      break;
   case 2:
      *red = 0.7;
      *grn = 0.7;
      *blu = 0.1;
      break;
   case 3:
      *red = 0.0;
      *grn = 0.5;
      *blu = 0.4;
      break;
   case 4:
      *red = 0.4;
      *grn = 0.4;
      *blu = 1.0;
      break;
   case 5:
      *red = 0.6;
      *grn = 0.6;
      *blu = 1.0;
      break;
   case 6:
      *red = 1.0;
      *grn = 0.6;
      *blu = 0.6;
      break;
   case 7:
      *red = 0.8;
      *grn = 0.4;
      *blu = 0.8;
      break;

   case 8:
      *red = 1.0;
      *grn = 0.3;
      *blu = 0.3;
      break;
   case 9:
      *red = 1.0;
      *grn = 0.5;
      *blu = 0.5;
      break;
   default:
      *red = 1.0;
      *grn = 0.5;
      *blu = 0.2;
      break;
}
```

After Modification:

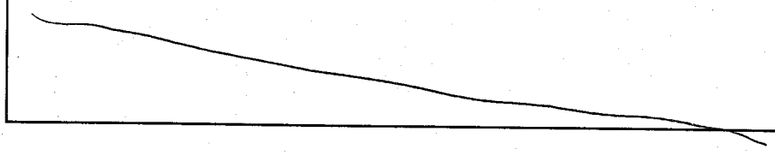


JWB


```
// Red to green
switch ((int)i) {
   case 0:
      *red = 1.0;
      *grn = 0.0;
      *blu = 0.0;
      break;
   case 1:
      *red = 1.0;
      *grn = 0.2;
      *blu = 0.0;
      break;
   case 2:
      *red = 1.0;
      *grn = 0.2;
      *blu = 0.0;
      break;
   case 3:
      *red = 1.0;
      *grn = 0.6;
      *blu = 0.0;
      break;
   case 4:
      *red = 1.0;
      *grn = 0.8;
      *blu = 0.0;
      break;
   case 5:
      *red = 1.0;
      *grn = 1.0;
      *blu = 0.0;
      break;
   case 6:
      *red = 0.8;
      *grn = 1.0;
      *blu = 0.0;
      break;
   case 7:
      *red = 0.6;
      *grn = 1.0;
      *blu = 0.7;
      break;
   case 8:
      *red = 0.4;
      *grn = 1.0;
      *blu = 1.0;
      break;
   case 9:
      *red = 0.2;
      *grn = 1.0;
      *blu = 0.4;
      break;
   default:
      *red = 0.0;
      *grn = 1.0;
      *blu = 0.0;
      break;
}
```

Purpose for Modification:

The original color scale was unnatural and did not form any sort of logical progression. The new color scale was implemented to remedy these problems.

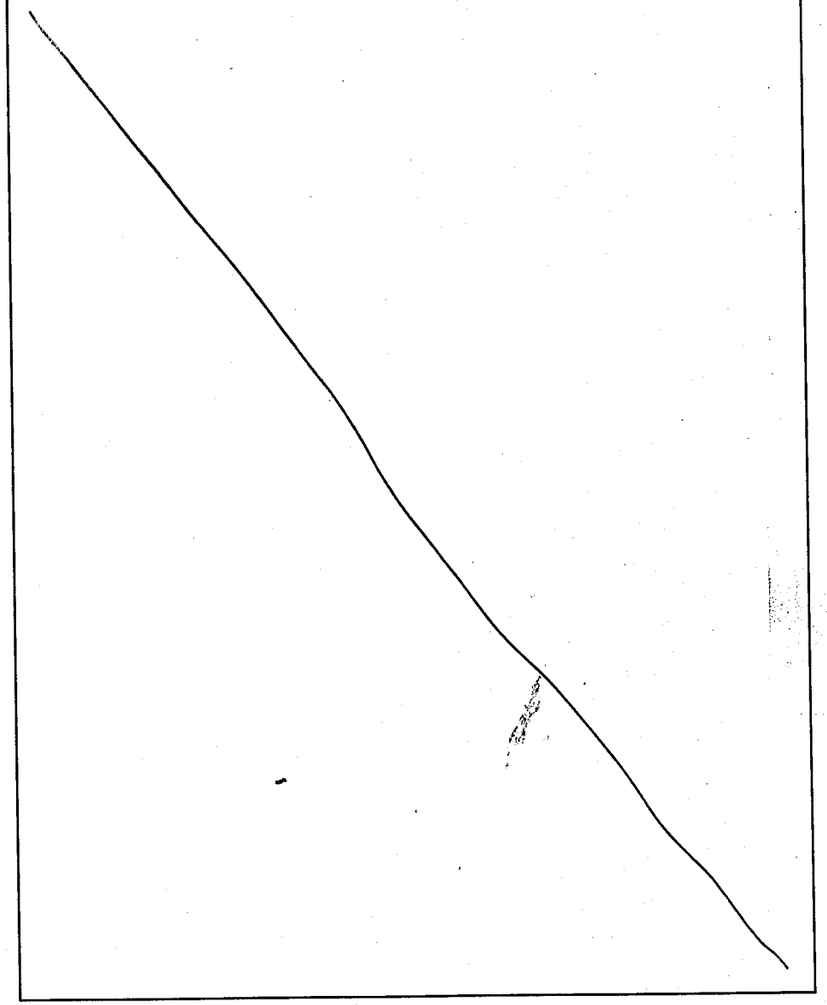


JWB

File Modified: ~3dstress/3d/help/MapViewer.sc

Purpose of Modification:

Wrote the documentation for the MapViewer of 3DStress using showcase to produce the on-line help file.



J W B

File: mapOptionCB.c++

Function Modified: void mapOptionWrite(Widget, XtPointer, XtPointer)

New Code:

```
void  mapOptionWrite(Widget, XtPointer, XtPointer) {
   MapAttributeType *mattr = mapObj.getAttributes();

   int            counter = 1, numPoints;
   XmString       titleStr;
   XmStringTable  strList;
   char           tmpStr[400];

   double **pointData;

   titleStr = XmStringCreateLocalized("File displayer");

   strList = (XmStringTable) XtMalloc((12+36000) * sizeof(XmString));
   if (strList == NULL) {
      fprintf(stderr,"Fatal error: unable to perform malloc");
      exit(20);
   }

   // Header
   sprintf(tmpStr, "#Created by: %s", ident);
   strList[counter++] = XmStringCreateLocalized(tmpStr);

   time_t t;
   char *timeStr;
   t = time(NULL);
   timeStr = asctime(localtime(&t));
   sprintf(tmpStr, "#Date: %s", timeStr);
   strList[counter++] = XmStringCreateLocalized(tmpStr);
   sprintf(tmpStr, "#Generated by: Map Viewer", timeStr);
   strList[counter++] = XmStringCreateLocalized(tmpStr);
   sprintf(tmpStr, "", timeStr);
   strList[counter++] = XmStringCreateLocalized(tmpStr);

   // Actual data
   sprintf(tmpStr, "#     X     Y     Az     Dip     Strike");
   strList[counter++] = XmStringCreateLocalized(tmpStr);

   numPoints = mapFiles.getNumPoints();

   pointData = (double **)XtMalloc(numPoints * sizeof(double *));

   if (pointData == NULL) {
      fprintf(stderr,"Fatal error: unable to perform malloc");
      exit(20);
   }

   for (int i = 0; i < numPoints; i++)
   {
      pointData[i] = (double *)XtMalloc(9 * sizeof(double));

      if (pointData[i] == NULL) {
         fprintf(stderr,"Fatal error: unable to perform malloc");
         exit(20);
      }
   }

   mapFiles.getPointData(pointData);

   for (i = 0; i < numPoints; i++) {
      sprintf(tmpStr,"%5.3lf %5.3lf %5.3lf %5.3lf %5.3lf %5.3lf",
         pointData[i][0], pointData[i][1], pointData[i][6],
         pointData[i][7], pointData[i][8]);

      strList[counter++] = XmStringCreateLocalized(tmpStr);
   } // end for loop
```


J W B

```
   fileShower(mattr->mapTop, strList, titleStr, counter);
} // end of showFile
```

File: fileSystem.hh

New Code: Added public methods to Class FileSystem:

```
    void getPointData(double **pointData);
    int  getNumPoints(void);
```

File: fileSystem.c++

Functions:

```
    void getPointData(double **pointData);
    int  getNumPoints(void);
```

New Code:

```
void FileSystem::getPointData(double **pointData)
{
   int startIndex;

   startIndex = 0;

    for (int i = 0; i < filesLoaded; i++)
        fileArray[i].getPointData(pointData, &startIndex);
}

int FileSystem::getNumPoints(void)
{
    int numPoints = 0;

    for (int i = 0; i < filesLoaded; i++)
        numPoints += fileArray[i].getNumPoints();

   return numPoints;
}
```

File: fileClass.hh

New Code: Added public methods to Class FileSystem:

```
      void       getPointData(double **pointData, int *startIndex);
      int        getNumPoints(void);
```

File: fileClass.c++

Functions:

```
      void       getPointData(double **pointData, int *startIndex);
      int        getNumPoints(void);
```

New Code:

```
void FileClass::getPointData(double **pointData, int *startIndex)
{

   for (int i = 0; i < num_of_segments; i++)
      segments[i].getPointData(pointData, startIndex);
}
```


J W B


```
int FileClass::getNumPoints(void)
{
    int numPoints = 0;

    for (int i = 0; i < num_of_segments; i++)
        numPoints += segments[i].getNumPoints();

   return numPoints;
}
```

File: segmentClass.hh

New Code: Added public methods to Class SegmentClass:

```
      void       getPointData(double **pointData, int *startIndex);
      int        getNumPoints(void);
```

File: segmentClass.c++

Functions:

```
      void       getPointData(double **pointData, int *startIndex);
      int        getNumPoints(void);
```

New Code:

```
void SegmentClass::getPointData(double **pointData, int *startIndex)
{
    for (int i = 0; i < number_of_points; i++)
    {
        pointData[i + *startIndex][0] = points[i].cord[0];
        pointData[i + *startIndex][1] = points[i].cord[1];
        pointData[i + *startIndex][2] = points[i].cord[2];
        pointData[i + *startIndex][3] = points[i].center[0];
        pointData[i + *startIndex][4] = points[i].center[1];
        pointData[i + *startIndex][5] = points[i].center[2];
        pointData[i + *startIndex][6] = points[i].az;
        pointData[i + *startIndex][7] = points[i].dip;
        pointData[i + *startIndex][8] = points[i].strike;
    }

    (*startIndex) += number_of_points;
}

int SegmentClass::getNumPoints(void)
{
    return number_of_points;
}
```

Purpose for Modifications:

All these modifications were made soley to produce a working browse-data window for the Map Viewer.

File: roseClass.c++

Function: void RoseClass::showFile(void)

New Code:

```
extern    float         lnCumLength[18];
extern    float         lnContLength[18];
```

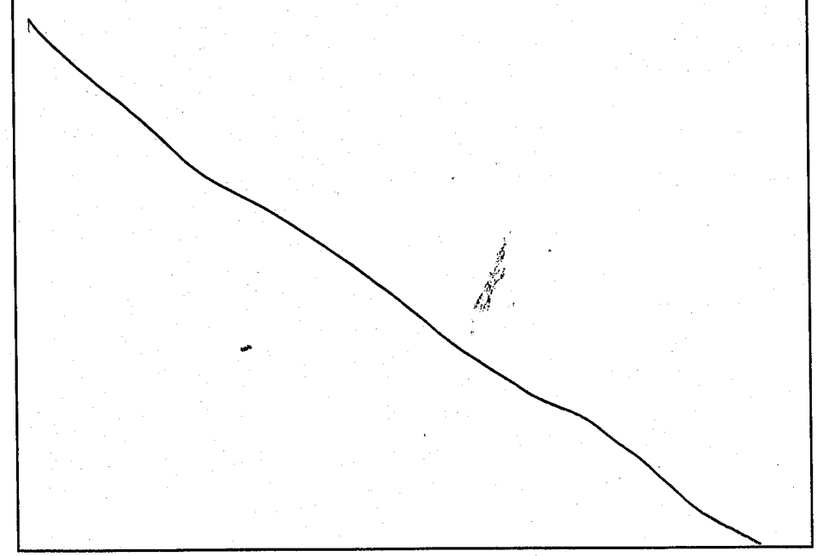
J W B

```
   if (!pattr.thresOption) {
      for (int i = 0; i < 180; i+=10) {
         sprintf(tmpStr,"%3d-%3d %12.6f %14.2f %14.2f",
                 i, i+10,
                 (float)plotTend[cind],
                 lnContLength[i/10], lnCumLength[i/10]);
         strList[counter++] = XmStringCreateLocalized(tmpStr);
         cind++;
         if (cind > 17) cind -= 18;
      } // end for int i
   } // end if
   else {
      double top =
         pattr.thresUpper * (dotDataMax-dotDataMin) + dotDataMin;
      double bottom =
         pattr.thresLower * (dotDataMax-dotDataMin) + dotDataMin;

         for (int i = 0; i < 180; i+=10) {
            if (plotTend[cind] >= bottom && plotTend[cind] <= top) {
               sprintf(tmpStr,"%3d-%3d %12.6f %14.2f %14.2f",
                       i, i+10,
                       (float)plotTend[cind],
                       lnContLength[i/10], lnCumLength[i/10]);
            } // end if
            strList[counter++] = XmStringCreateLocalized(tmpStr);
            cind++;
            if (cind > 17) cind -= 18;
         } // end for int i
   } // end else
```

==========

Purpose for Modifications:

The modifications allow the user to browse the data by clicking on the Browse Data Button in the Rose Options Window.



J W B

File Modified: viewerClass.c++

Function Modified: void ViewerClass::showFile (Widget w, int showCenter)

Resulting Code:

```
void ViewerClass::showFile (Widget w, int showCenter) {

   int
      i, counter = 0, numPoints = 0,
      tdir, tplunge,
      tmp = 1;

   char tmpStr[500];

   double **pointData;

   XmString        titleStr;
   XmStringTable   strList;

   PlotAttrType pattr = plotObj.getAttributes();

   titleStr = XmStringCreateLocalized("File displayer");

   numPoints = viewerFiles.getNumPoints();

   pointData = (double **)XtMalloc(numPoints * sizeof(double *));

   if (pointData == NULL) {
      fprintf(stderr,"Fatal error: unable to perform malloc");
      exit(20);
   }

   for (i = 0; i < numPoints; i++){
      pointData[i] = (double *)XtMalloc(9 * sizeof(double));

      if (pointData[i] == NULL) {
         fprintf(stderr,"Fatal error: unable to perform malloc");
         exit(20);
      }
   }

   viewerFiles.getPointData(pointData);

   counter += numPoints * 2;

   strList = (XmStringTable) XtMalloc(counter * sizeof(XmString));
   if (strList == NULL) {
      fprintf(stderr,"Fatal error: unable to perform malloc");
      exit(17);
   }
   // Header
   sprintf(tmpStr, "#Created by: %s", ident);
   strList[tmp++] = XmStringCreateLocalized(tmpStr);

   time_t t;
   char *timeStr;
   t = time(NULL);
   timeStr = asctime(localtime(&t));
   sprintf(tmpStr, "#Date: %s", timeStr);
   strList[tmp++] = XmStringCreateLocalized(tmpStr);
   sprintf(tmpStr, "#Generated by: 3D Fault Viewer");
   strList[tmp++] = XmStringCreateLocalized(tmpStr);
   sprintf(tmpStr, "");
   strList[tmp++] = XmStringCreateLocalized(tmpStr);
   sprintf(tmpStr, "#    Mag     Dir    Plng");
   strList[tmp++] = XmStringCreateLocalized(tmpStr);

   tdir    = (int)rint(pattr.sxdir);
   tplunge = (int)rint(pattr.sxplunge);
   if (tplunge < 0) {
```


J W B

```
      tdir += 180;
      if (tdir > 360) tdir -= 360;
      tplunge *= -1;
   }
   sprintf(tmpStr, "#SU  %3d %3d %3d", (int)(plotObj.getStressX()),
           tdir, tplunge);
   strList[tmp++] = XmStringCreateLocalized(tmpStr);
   tdir    = (int)rint(pattr.sydir);
   tplunge = (int)rint(pattr.syplunge);
   if (tplunge < 0) {
      tdir += 180;
      if (tdir > 360) tdir -= 360;
      tplunge *= -1;
   }
   sprintf(tmpStr, "#SV  %3d %3d %3d", (int)(plotObj.getStressY()),
           tdir, tplunge);
   strList[tmp++] = XmStringCreateLocalized(tmpStr);

   tdir    = (int)rint(pattr.szdir);
   tplunge = (int)rint(pattr.szplunge);
   if (tplunge < 0) {
   tdir += 180;
      if (tdir > 360) tdir -= 360;
      tplunge *= -1;
   }
   sprintf(tmpStr, "#SW  %3d %3d %3d", (int)(plotObj.getStressZ()),
           tdir, tplunge);
   strList[tmp++] = XmStringCreateLocalized(tmpStr);

   sprintf(tmpStr, "");
   strList[tmp++] = XmStringCreateLocalized(tmpStr);

   int count;
   float tmpAz, tmpPlng;

   if (!pattr.thresOption) {

      if (pattr.displayMode == StressRatio) {

         if (!showCenter)
            strList[tmp++] = XmStringCreateLocalized
                             ("#X-Y-Z-Slip_Tendency-Slip_AZ-Slip_Plng");
         else
            strList[tmp++] = XmStringCreateLocalized
                             ("#(Center) X-Y-Z-Slip_Tendency-Slip_AZ-Slip_Plng");
         count = 0;

         strList[tmp++] = XmStringCreateLocalized("");
         strList[tmp++] = XmStringCreateLocalized("#-=-=-=-=-=");

         for (i = 0; i < numPoints; i++) {

            if (count == 0) {

               plotObj.findSlipV(pointData[i][6], pointData[i][7]);

               tmpAz = (float)plotObj.getslipVAz();
               tmpPlng = (float)plotObj.getslipVPlunge();

               if (!showCenter) {
                  sprintf(tmpStr,"%E %E %E %f %f %f",
                     pointData[i][0], pointData[i][1], pointData[i][2],
                     (float)plotObj.getTendency(),
                     tmpAz, tmpPlng);
               }
               else {
                  sprintf(tmpStr,"%E %E %E %f %f %f",
                     pointData[i][3], pointData[i][4], pointData[i][5],
                     (float)plotObj.getTendency(),
                     tmpAz, tmpPlng);
               }
```

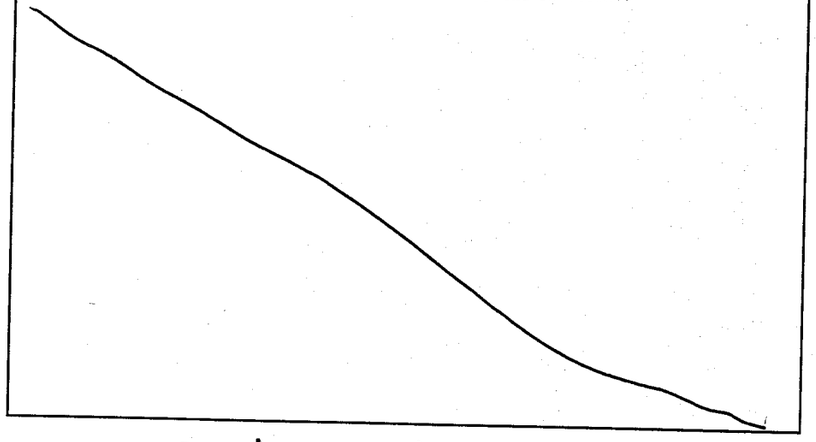


```
               strList[tmp++] = XmStringCreateLocalized(tmpStr);
            }
            else if (!showCenter){

               sprintf(tmpStr,"%E %E %E NULL NULL NULL",
                  pointData[i][0], pointData[i][1], pointData[i][2]);

               strList[tmp++] = XmStringCreateLocalized(tmpStr);

               if (count == 2)
                  count = -1;
            }
            else if (count == 2)
               count = -1;

            count++;
         } // end of for i from 0 to numPoints
      } // endif stressRatio
      else {

         if (!showCenter)
            strList[tmp++] = XmStringCreateLocalized
                             ("#X-Y-Z-Dilation_Tendency");
         else
            strList[tmp++] = XmStringCreateLocalized
                             ("#(Center) X-Y-Z-Dilation_Tendency");
         count = 0;

         strList[tmp++] = XmStringCreateLocalized("");
         strList[tmp++] = XmStringCreateLocalized("#-=-=-=-=-=");

         for (i = 0; i < numPoints; i++) {

            if (count == 0) {

               plotObj.findSlipV(pointData[i][6], pointData[i][7]);

               if (!showCenter)
                  sprintf(tmpStr,"%E %E %E %f",
                     pointData[i][0], pointData[i][1], pointData[i][2],
                     (float)plotObj.getDilationTend());
               else
                  sprintf(tmpStr,"%E %E %E %f",
                     pointData[i][3], pointData[i][4], pointData[i][5],
                     (float)plotObj.getDilationTend());

               strList[tmp++] = XmStringCreateLocalized(tmpStr);
            }
            else if (!showCenter){

               sprintf(tmpStr,"%E %E %E NULL",
                  pointData[i][0], pointData[i][1], pointData[i][2]);

               strList[tmp++] = XmStringCreateLocalized(tmpStr);

               if (count == 2)
                  count = -1;
            }

            else if (count ==2)
               count = -1;

            count++;
         } // endfor i
      } // end else
   } // endif pattr.thresOption
   else {

      double top =
         pattr.thresUpper * (dotDataMax-dotDataMin) + dotDataMin;
      double bottom =
```

```
      pattr.thresLower * (dotDataMax-dotDataMin) + dotDataMin;
   float tmpTend;

   if (pattr.displayMode == StressRatio) {
      if (!showCenter)
         strList[tmp++] = XmStringCreateLocalized
                          ("#X-Y-Z-Slip_Tendency-Slip_AZ-Slip_Plng");
      else
         strList[tmp++] = XmStringCreateLocalized
                          ("#(Center) X-Y-Z-Slip_Tendency-Slip_AZ-Slip_Plng");

      strList[tmp++] = XmStringCreateLocalized("");
      strList[tmp++] = XmStringCreateLocalized("#-=-=-=-=-=");

      for (i = 0; i < numPoints; i++) {

         plotObj.findSlipV(pointData[i][6], pointData[i][7]);

         tmpAz = (float)plotObj.getslipVAz();
         tmpPlng = (float)plotObj.getslipVPlunge();
         tmpTend = (float)plotObj.getTendency();

         if (tmpTend >= bottom && tmpTend <= top) {

            if (!showCenter) {

               sprintf(tmpStr,"%E %E %E %f %f %f",
                  pointData[i][0], pointData[i][1], pointData[i][2],
                  tmpTend, tmpAz, tmpPlng);

               strList[tmp++]=XmStringCreateLocalized(tmpStr);

               sprintf(tmpStr,"%E %E %E NULL NULL NULL",
                  pointData[i][0], pointData[i][1], pointData[i][2]);

               strList[tmp++]=XmStringCreateLocalized(tmpStr);

               sprintf(tmpStr,"%E %E %E NULL NULL NULL",
                  pointData[i][0], pointData[i][1], pointData[i][2]);

               strList[tmp++]=XmStringCreateLocalized(tmpStr);
            } // end if(!showCenter)
            else {

               sprintf(tmpStr,"%E %E %E %f %f %f",
                  pointData[i][3], pointData[i][4], pointData[i][5],
                  tmpTend, tmpAz, tmpPlng);

               strList[tmp++]=XmStringCreateLocalized(tmpStr);
            } // end else
         } // end if (tmpTend >= bottom && tmpTend <= top)
      } // endfor i
   } // end if pattr.displayMode
   else {

      if (!showCenter)
         strList[tmp++] = XmStringCreateLocalized
                          ("#X-Y-Z-Dilation_Tendency");
      else
         strList[tmp++] = XmStringCreateLocalized
                          ("#(Center) X-Y-Z-Dilation_Tendency");

      strList[tmp++] = XmStringCreateLocalized("");
      strList[tmp++] = XmStringCreateLocalized("#-=-=-=-=-=");

      for (i = 0; i < numPoints; i++) {

         plotObj.findSlipV(pointData[i][6], pointData[i][7]);

         tmpTend = plotObj.getDilationTend();
```


J W B


```
            if (tmpTend >= bottom && tmpTend<= top) {

               if (!showCenter) {

                  sprintf(tmpStr,"%E %E %E %f",
                     pointData[i][0], pointData[i][1], pointData[i][2],
                     tmpTend);

                  strList[tmp++]=XmStringCreateLocalized(tmpStr);

                  sprintf(tmpStr,"%E %E %E NULL",
                     pointData[i][0], pointData[i][1], pointData[i][2]);

                  strList[tmp++]=XmStringCreateLocalized(tmpStr);

                  sprintf(tmpStr,"%E %E %E NULL",
                     pointData[i][0], pointData[i][1], pointData[i][2]);

                  strList[tmp++]=XmStringCreateLocalized(tmpStr);
               } // end if (!showCenter)
               else {

                  sprintf(tmpStr,"%E %E %E %f",
                     pointData[i][3], pointData[i][4], pointData[i][5],
                     tmpTend);

                  strList[tmp++]=XmStringCreateLocalized(tmpStr);
               } // end else
            } // end if (tmpTend >= bottom && tmpTend<= top)
         } // end for (i = 0; i < numPoints; i++)
      } // endelse

      counter = tmp;
   } // end else

   fileShower(w, strList, titleStr, tmp);

} // end of showFile

Purpose for Modifications:

        The modifications to showFile allow the browse data button in the Viewer
Options window to make use of the new file system.  The browse data feature
now allows the user to see point data produced by the 3d Viewer.
```



J W B

May 25, 98 14:20 **08Jan98.log** Page 1/1

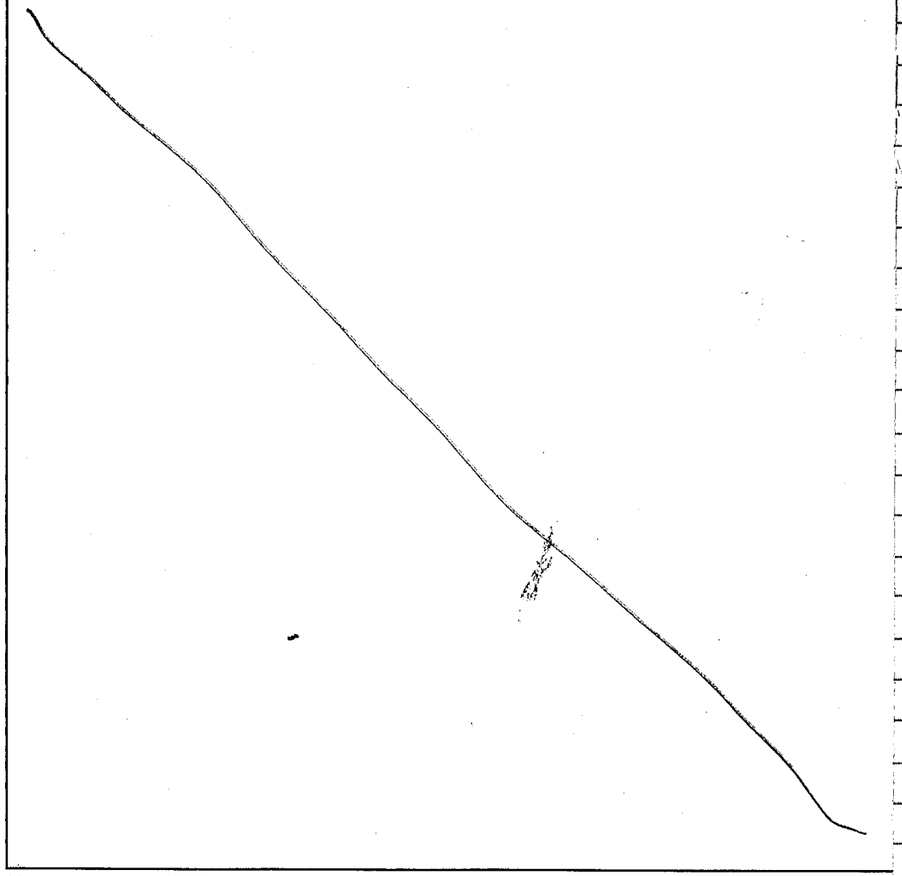
Files Modified:

mohrOptionCB.c++
mohrOptionCB.hh
mohrOptionClass.c++
mohrOptionClass.hh

Purpose for Modification:

A button was added in the Mohr Graph Options Window that when clicked presents a pop-up information window describing the units of measure used in the Mohr Graph calculations.

Also, the Rock Info citations were corrected and expanded.



Monday May 25, 1998 ../Notes/

JWB

May 25, 98 14:20 **09March98.log** Page 1/1

File Changed: optionCallbacks.c++

Function Changed:

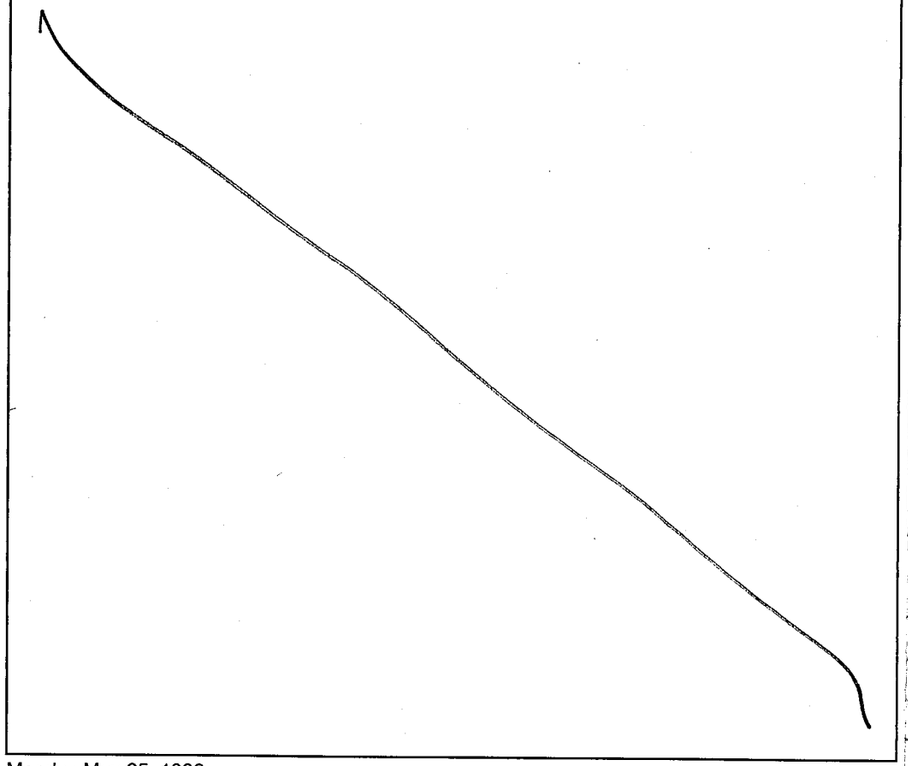
void changeStepSize(Widget, XtPointer client_data, XtPointer)

Reason for change:

A change in step size previously reset the map's fault dip slider to zero. Now, a change in step size preserves the fault dip value modulo the new step size.

The following files were collated with existing 3dstress inner workings for the variable stress field project:

cmdClass.c++	optionCallbacks.c++	plotClass.hh
cmdClass.hh	optionCallbacks.hh	plotClass_display.c++
cmdClass_magDialog.c++	optionClass.c++	plotClass_findSlipV.c++
cmdClass_scaleCB.c++	optionClass.hh	
cmdClass_toggleCB.c++	plotClass.c++	



Monday May 25, 1998 ../Notes/09

JWB

May 25, 98 14:20 **03April98.log** Page 1/1

File Modified: optionCallbacks.c++

Function Modified: void changeStepSize(Widget, XtPointer client_data, XtPointer
)

Purpose:

When the Plot scale was changed from the Options window, no check was made of the mapOption form widget. This resulted in a segmentation fault if the map option window had not been previously openned. A test was added before any change made to the mapOptionObj.atributes.mapScalePlungeAngle

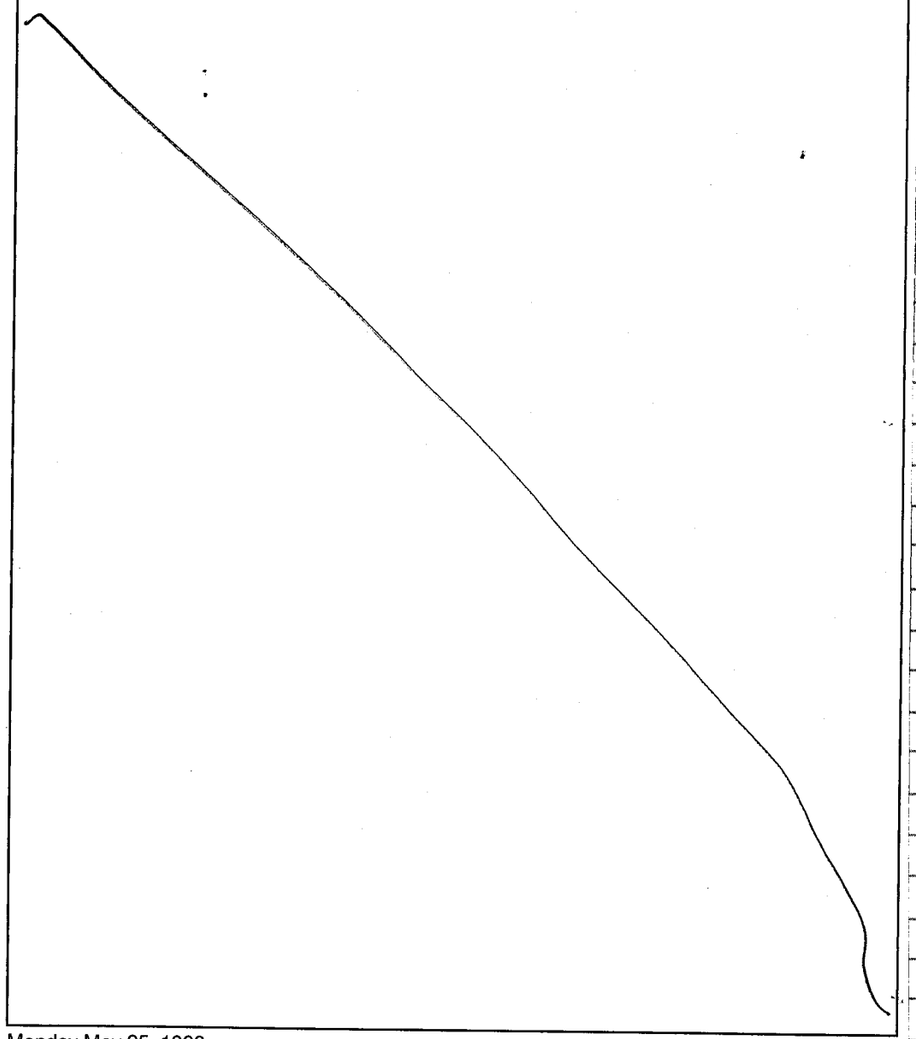
Modified Code:

```
   switch ((int)client_data) {
      case 0:
         if (mwattr->mapScalePlungeAngle && plotAttr.azColorMode != Max) {

                                 tmpMapScaleValue -= tmpMapScaleValue % 10;
                                 tmpPlotPlungeAngle -= (float) (((int) tmpPlotPlu
ngeAngle) % 10);

===>        if(mwattr->mapDialogForm != NULL)  // check to see if map option win
dow exists
            {
                XtVaSetValues(mwattr->mapScalePlungeAngle,
                   XmNscaleMultiple, 10,
                   XmNvalue, tmpMapScaleValue, NULL);
            }
            plotObj.setPlungeAngle(tmpPlotPlungeAngle);
         }
         plotAttr.azStepSize = 10.0;
         plotAttr.plungeStepSize = 10.0;
         break;
      case 1:
         if (mwattr->mapScalePlungeAngle && plotAttr.azColorMode != Max) {

                                 tmpMapScaleValue -= tmpMapScaleValue % 5;
                                 tmpPlotPlungeAngle -= (float) (((int) tmpPlotPlu
ngeAngle) % 5);
===>        if(mwattr->mapDialogForm != NULL)  // check to see if map option win
dow exists
            {
                XtVaSetValues(mwattr->mapScalePlungeAngle,
                   XmNscaleMultiple, 5,
                   XmNvalue, tmpMapScaleValue, NULL);
            }
            plotObj.setPlungeAngle(tmpPlotPlungeAngle);
         }
         plotAttr.azStepSize = 5.0;
         plotAttr.plungeStepSize = 5.0;
         break;
      case 2:
         plotAttr.azStepSize = 1.0;
         plotAttr.plungeStepSize = 1.0;
         break;
      default:
         break;
```


J W B

May 25, 98 14:20 **27April98.log** Page 1/1

Proposal:
in the file plotClass_display.c++, near line 1191, change the displayed equations to make use of the sigma symbol instead of writing out the words. Just use the plotFont.drawGLSymbol function as in the case of the 3rd stress plot type.



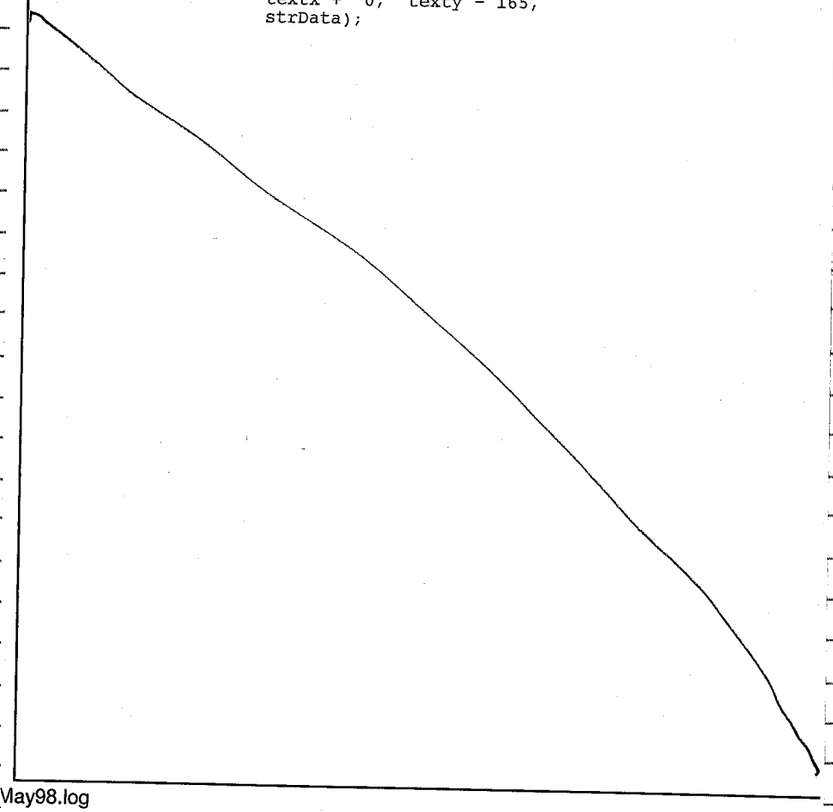
J W B

```
File Name:  plotClass_display.c++

Function:  PlotClass.display

Purpose for modification:
  in the file plotClass_display.c++, near line 1191,
  changed the displayed equations to make use of the
  sigma symbol instead of writing out the words.  Just
  used the plotFont.drawGLSymbol function as in the case
  of the 3rd stress plot type.

Modified Code:

      if (attributes.displayMode == StressRatio) {
         plotFont.drawGLSymbol(attributes.textred,
                               attributes.textgrn,
                               attributes.textblu,
                               135,125,
                               "s" );
         plotFont.drawGLFont(attributes.textred,
                  attributes.textgrn,
                  attributes.textblu,
                   141, 125,
                  "S");
         plotFont.drawGLSymbol(attributes.textred,
                               attributes.textgrn,
                               attributes.textblu,
                               146,125,
                               "/sN" );
      } else if (attributes.displayMode == Fluid) {
         plotFont.drawGLFont(attributes.textred,
                  attributes.textgrn,
                  attributes.textblu,
                   130, 125,
                  "pf");
            plotFont.drawGLSymbol(attributes.textred,
                               attributes.textgrn,
                               attributes.textblu,
                               140,125,
                               "/(sN-T)" );
      }

=====================================================================

File Name:  plotClass_display.c++

Function:  PlotClass.display

Purpose for modification:
  Rake was set to and displayed as a nan when strike and/or dip were at
  extrema.  Now rake is displayed as 'undefined' in this situation.

Modified Code:

      //-------------------------------------------------
      // Calculate Rake
      //-------------------------------------------------
      double tmpDenom, tmpRake;
      int tmpErrFlag = 0;

      tmpRake = (fltva[0] * slipVX + fltva[1] * slipVY);
      tmpDenom = ( sqrt(fltva[0] * fltva[0] + fltva[1] * fltva[1]) *
```


J W B

Printed by Joshu

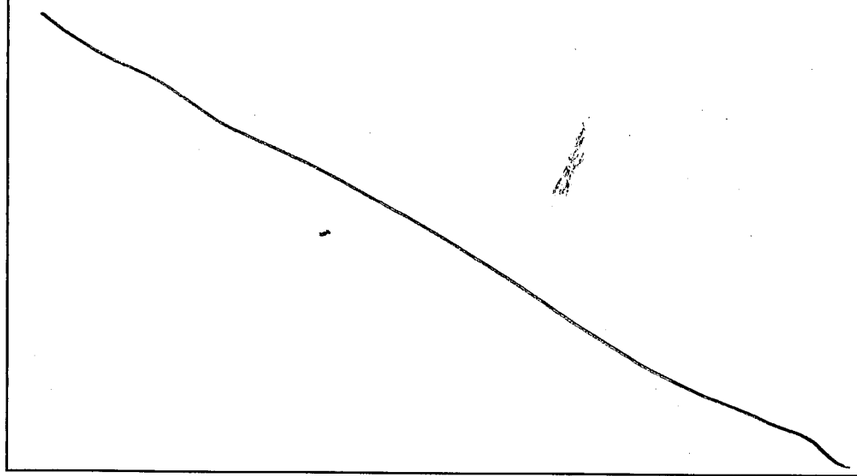

```
                  sqrt(slipVX * slipVX + slipVY * slipVY));

      if (tmpDenom == 0.0)
        tmpErrFlag = 1;
      else
        tmpRake = tmpRake / tmpDenom;
<snip>

            //-------------------------------------
            // Rake
            //-------------------------------------
            if(tmpErrFlag != 1)
              sprintf(strData, "Rake          %4.1f", (float)tmpRake);
            else
              sprintf(strData, "Rake          undefined");
            drawText(attributes.textred,
                     attributes.textgrn,
                     attributes.textblu,
                     textx +  0,  texty - 165,
                     strData);
```



J W B

May 25, 98 14:20 **13May98.log** Page 1/1

```
Completed task #7 on the 3dstress 1.3 SRD by applying the UNIX command
grep -n helpObj <filename list>
to all major window classes and checking for two relivant finds.  Everything
was already coded correctly, so no changes were made.

================================================================================

Files Modified:

  Makefile
  mapButtonCB.c++
  mapButtonCB.hh
  mapClass.c++
  mapOptionCB.c++
  mapOptionCB.hh
  mapOptionClass.c++
  mapOptionClass.hh
  viewNetGlobals.hh

Files Created:

  lineBuilderCB.c++
  lineBuilderClass.c++
  lineBuilderObj.hh
  lineBuilderCB.hh
  lineBuilderClass.hh

Purpose for Modifications:

  Attempted to produce a new, separate window for the line building
  features of the Map Viewer.  Simply mimicked the faultBuilder
  class structure and pasted in existing map-specific functions.  Initial
  results promising.  Should finish by the end of 14 May 1998.  All that
  is left to do is to add the new files to RCS.  This completes task # 54
  of the 3dstress 1.3 SRD.
```



Monday May 25, 1998 ../Notes/1

J W B

May 25, 98 14:20 **14May98.log** Page 1/3

```
Files Modified:

/usr/people/3dstress/3d/src/lineBuilderCB.c++
/usr/people/3dstress/3d/src/lineBuilderCB.hh
/usr/people/3dstress/3d/src/lineBuilderClass.c++
/usr/people/3dstress/3d/src/lineBuilderClass.hh
/usr/people/3dstress/3d/src/lineBuilderObj.hh

Modification description:

Fixed up the File Header necessary for RCS.

Example Header:

//////////////////////////////////////////////////
//  Filename:    lineBuilderObj.hh
//  Author:      Joshua Buckner
//  Date:        05-14-98
//
//    Developed by the Center for Nuclear Waste Regulatory
//    Analyses (CNWRA), Southwest Research Institute (SwRI),
//    San Antonio, Texas, USA.
//
//    Copyright 07/20/95 Southwest Research Institute
//                All rights reserved.
//
//    This software is a trade secret owned by Southwest Research
//    Institute, with access limited except as required for use by
//    authorized users.
//
//    This program was developed under sponsorship of the U.S.
//    Nuclear Regulatory Commission, contract number NRC-02-93-005.
//
//
//  Purpose:
//  Map Builder Gui.
//
//  $Header: /usr/people/3dstress/3d/src/RCS/lineBuilderObj.hh,v 1.3 1998/05/14
14:31:54 jbuckner Exp $
//
//  Revision History
//  $Log: lineBuilderObj.hh,v $
//  Revision 1.3  1998/05/14  14:31:54  jbuckner
//  fixed file header
//
//  Revision 1.1  1998/05/14  09:13:07  jbuckner
//  Initial revision
//
//
//
//////////////////////////////////////////////////

================================================================================

File Modified:  optionClass.hh

Description of Modification:

  Added the following Widgets to the attributes structure of
  the OptionClass class:

  controlOrientRadio,
```

Monday May 25, 1998 ../Notes/1

J W B

Printed by Joshua


```
  controlOrientRadio1,
  controlOrientRadio2.

  These were used to create radio buttons enabling the user to
  switch the control bar's orientation between horizontal and
  vertical.

File Modified:  optionClass.c++

Function Modified:  OptionClass:initialize

Added Code:

      Widget
              col6,
              controlOrientLabel;
      XmString
              controlOrientStr,
              controlOrientStr1,
              controlOrientStr2,

      .
      .
      .

      controlOrientStr = XmStringCreateLocalized("Control Bar Orientation
      controlOrientStr1 = XmStringCreateLocalized("Horizontal");
      controlOrientStr2 = XmStringCreateLocalized("Vertical");

      .
      .
      .

      frame = XtVaCreateManagedWidget("frame", xmFrameWidgetClass,
                                      colWidget,
                                      XmNbackground, WIDGET_COLOR, NULL);


      col6 = XtVaCreateWidget (
           "col6",
           xmRowColumnWidgetClass,
           frame,
           XmNorientation, XmVERTICAL,
           XmNnumColumns, 2,
           XmNpacking, XmPACK_TIGHT,
           XmNbackground, WIDGET_COLOR,
           NULL);

      controlOrientLabel = XtVaCreateManagedWidget("commandOrientLabel",
                                          xmLabelWidgetClass,
                                          col6,
                                          XmNbackground, WIDGET_COLOR,
                                          XmNalignment,
                                          XmALIGNMENT_BEGINNING,
                                          XmNlabelString, controlOrien
                                          NULL);


      n = 0;
      XtSetArg(args[n], XmNorientation, XmHORIZONTAL); n++;

      attributes.controlOrientRadio = XmCreateRadioBox(col6, "controlOrie
", args, n);
```


JWB


```
      XmChangeColor(attributes.controlOrientRadio, WIDGET_COLOR);

      attributes.controlOrientRadio1 = XtVaCreateManagedWidget("controlOrientRad
io1",
                                          xmToggleButtonGadgetClass,
                                          attributes.controlOrientRadio,
                                          XmNlabelString, controlOrientStr1,
                                          XmNselectColor, BUTTON_COLOR,
                                          NULL);

      XmToggleButtonSetState(attributes.controlOrientRadio1, TRUE, TRUE);

      attributes.controlOrientRadio2 = XtVaCreateManagedWidget("controlOrientRad
io2",
                                          XmNlabelString, controlOrientStr2,
                                          XmNselectColor, BUTTON_COLOR,
                                          NULL);
      .
      .
      .


      XtManageChild(attributes.controlOrientRadio);
      XtManageChild(col6);                                        xmToggleBut
tonGadgetClass,

      .
      .
      .


      XmStringFree(fluidStr);
      XmStringFree(verboseStr);
      XmStringFree(colorStr);
      XmStringFree(plotStr);
      XmStringFree(plotColorStr);
      XmStringFree(bWStr);
      XmStringFree(backStr);
      XmStringFree(blackStr);
      XmStringFree(whiteStr);
      XmStringFree(controlOrientStr);
      XmStringFree(controlOrientStr1);
      XmStringFree(controlOrientStr2);
attributes.controlOrientRadio,

Purpose for Modification:

  To create radio buttons enabling the user to
  switch the control bar's orientation between horizontal and
  vertical.  Also, made sure that all strings allocated were
  freed.  The necessary callbacks (to be added to optionCallbacks)
  remain for 15 May 1998 as does the necessary facility within the
  ControlClass.
```


JWB

```
File Modified: optionClass.c++

Code Added:

      XtAddCallback(attributes.controlOrientRadio1, XmNvalueChangedCallback,
                    changeControlOrient, (XtPointer)1);
      XtAddCallback(attributes.controlOrientRadio2, XmNvalueChangedCallback,
                    changeControlOrient, (XtPointer)2);

Purpose:

  Added callbacks to change the orientation of the control bar from
  horizontal to veritcal and vis versa.

File Modified: optionCallbacks.hh

Code Added:

  void changeControlOrient(Widget, XtPointer, XtPointer);

Purpose:

  Added callbacks to change the orientation of the control bar from
  horizontal to veritcal and vis versa.

File Modified: optionCallbacks.c++

Code Added:

void changeControlOrient(Widget, XtPointer client_data, XtPointer)
{
  if ((int)client_data == 1)
    cntrlObj.horizontal();
  else
    cntrlObj.vertical();
}

Purpose:

  Added callbacks to change the orientation of the control bar from
  horizontal to veritcal and vis versa.

File Modified: controlClass.hh

Code Added:

   public:
      void   horizontal(void);
      void   vertical(void);
   private:
      Widget
        parent, form, buttons[10];

Purpose:

  Added members and functions to facilitate the change of orientation from
  horizontal to veritcal and vis versa.

File Modified: controlClass.c++

Code Added:
```


JWB

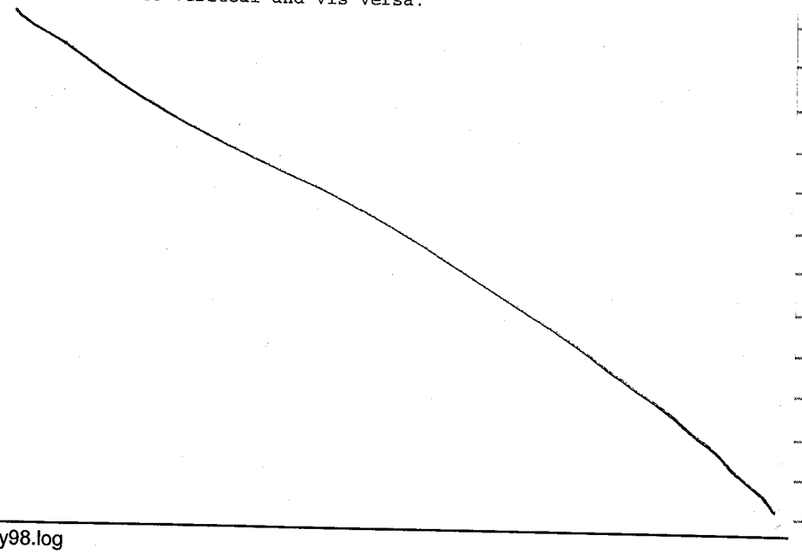

```
void ControlClass::horizontal(void) {
   if(parent != NULL)
   {
      XtVaSetValues(parent, XmNmaxHeight, 50, NULL);
      XtVaSetValues(parent, XmNheight, 50, NULL);
      XtVaSetValues(parent, XmNmaxWidth, 900, NULL);
      XtVaSetValues(parent, XmNwidth, 900, NULL);
      XtDestroyWidget(form);
      initialize(parent);
   }
}

void ControlClass::vertical(void) {
   int       n,
             topPos = 1,
             bottomPos = 5;
   Arg       args[12];
   Pixmap pixmap[10];

   if(parent == NULL)
     return;

   XtDestroyWidget(form);

   XtVaSetValues(parent, XmNmaxHeight, 900, NULL);
   XtVaSetValues(parent, XmNheight, 900, NULL);
   XtVaSetValues(parent, XmNmaxWidth, 50, NULL);
   XtVaSetValues(parent, XmNwidth, 50, NULL);

   attributes.display = YesC;
   n = 0;
   XtSetArg(args[n], XmNmaxWidth, 50); n++;
   XtSetArg(args[n], XmNwidth, 50); n++;
   XtSetArg(args[n], XmNmaxHeight, 353); n++;
   XtSetArg(args[n], XmNheight, 353); n++;
   XtSetArg(args[n], XmNbackground, WIDGET_COLOR); n++;
   XtSetArg(args[n], XmNtraversalOn, TRUE); n++;
   XtSetArg(args[n], XmNfractionBase, 71); n++;
   XtSetArg(args[n], XmNnoResize, TRUE); n++;


   form = XmCreateForm(parent,
                       "form",
                       args, n);
   for (int i = 0; i < 10; i++) {
      buttons[i] = XtVaCreateManagedWidget("button",
                                           xmPushButtonWidgetClass,
                                           form,
                                           XmNnoResize, TRUE,
                                           XmNrightAttachment,
                                           XmATTACH_FORM,
                                           XmNleftAttachment,
                                           XmATTACH_FORM,
                                           XmNbottomAttachment,
                                           XmATTACH_POSITION,
                                           XmNtopAttachment,
                                           XmATTACH_POSITION,

                                           XmNtopPosition, topPos,
                                           XmNbottomPosition, bottomPos,
                                           NULL);
      topPos += 5;
```


JWB

```
    bottomPos += 5;
  } // endfor int i

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(topLevel)),
                        plot8_xpm, &pixmap[0], NULL, NULL);
    XtVaSetValues(buttons[0],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[0],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        sliders8_xpm, &pixmap[1], NULL, NULL);
    XtVaSetValues(buttons[1],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[1],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        graph8_xpm, &pixmap[2], NULL, NULL);
    XtVaSetValues(buttons[2],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[2],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        mohr8_xpm, &pixmap[3], NULL, NULL);
    XtVaSetValues(buttons[3],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[3],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        dview8_xpm, &pixmap[4], NULL, NULL);
    XtVaSetValues(buttons[4],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[4],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        map8_xpm, &pixmap[5], NULL, NULL);
    XtVaSetValues(buttons[5],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[5],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        surf8_xpm, &pixmap[6], NULL, NULL);
    XtVaSetValues(buttons[6],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[6],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        option8_xpm, &pixmap[7], NULL, NULL);
```



```
    XtVaSetValues(buttons[7],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[7],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        exit8_xpm, &pixmap[8], NULL, NULL);
    XtVaSetValues(buttons[8],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[8],
                  NULL);

    XpmCreatePixmapFromData(XtDisplay(topLevel),
                        XRootWindowOfScreen(XtScreen(form)),
                        help8_xpm, &pixmap[9], NULL, NULL);
    XtVaSetValues(buttons[9],
                  XmNlabelType, XmPIXMAP,
                  XmNlabelPixmap, pixmap[9],
                  NULL);

  for (i = 0; i < 10; i++) {
    XtAddCallback(buttons[i], XmNactivateCallback,
                  controlerCB, (XtPointer)i);
//    XFreePixmap(XtDisplay(topLevel), pixmap[i]);

  } // endfor i = 0
  XtManageChild(form);
}

Purpose:

  Added members and functions to facilitate the change of orientation fro
  horizontal to veritcal and vis versa.
```



May 25, 98 14:20 **22May98.log** Page 1/1

File Modified: controlClass.c++

Function Modified: void ControlClass::vertical(void)

Modified Lines:

```
   int      n,
            topPos = 1,
            bottomPos = 6;

   .
   .
   .

   XtVaSetValues(parent, XmNmaxHeight, 700, NULL);
   XtVaSetValues(parent, XmNheight, 700, NULL);
   XtVaSetValues(parent, XmNmaxWidth, 70, NULL);
   XtVaSetValues(parent, XmNwidth, 70, NULL);

   .
   .
   .

   XtSetArg(args[n], XmNmaxWidth, 60); n++;
   XtSetArg(args[n], XmNwidth, 60); n++;
   XtSetArg(args[n], XmNmaxHeight, 353); n++;
   XtSetArg(args[n], XmNheight, 353); n++;

   .
   .
   .

  for (int i = 0; i < 10; i++) {
     buttons[i] = XtVaCreateManagedWidget("button",
                                          xmPushButtonWidgetClass,
                                          form,
                                          XmNnoResize, TRUE,
                                          XmNrightAttachment,
                                          XmATTACH_POSITION,
                                          XmNleftAttachment,
                                          XmATTACH_POSITION,
                                          XmNbottomAttachment,
                                          XmATTACH_POSITION,
                                          XmNtopAttachment,
                                          XmATTACH_POSITION,
                                          XmNleftPosition, 7,
                                          XmNrightPosition, 57,
                                          XmNtopPosition, topPos,
                                          XmNbottomPosition, bottomPos,
                                          NULL);
     topPos += 7;
     bottomPos = topPos + 5;
```

Purpose for Modification:

Finishing up last project: made the buttons fit in the vertical control bar in an asthetically pleasing manner.

JWB

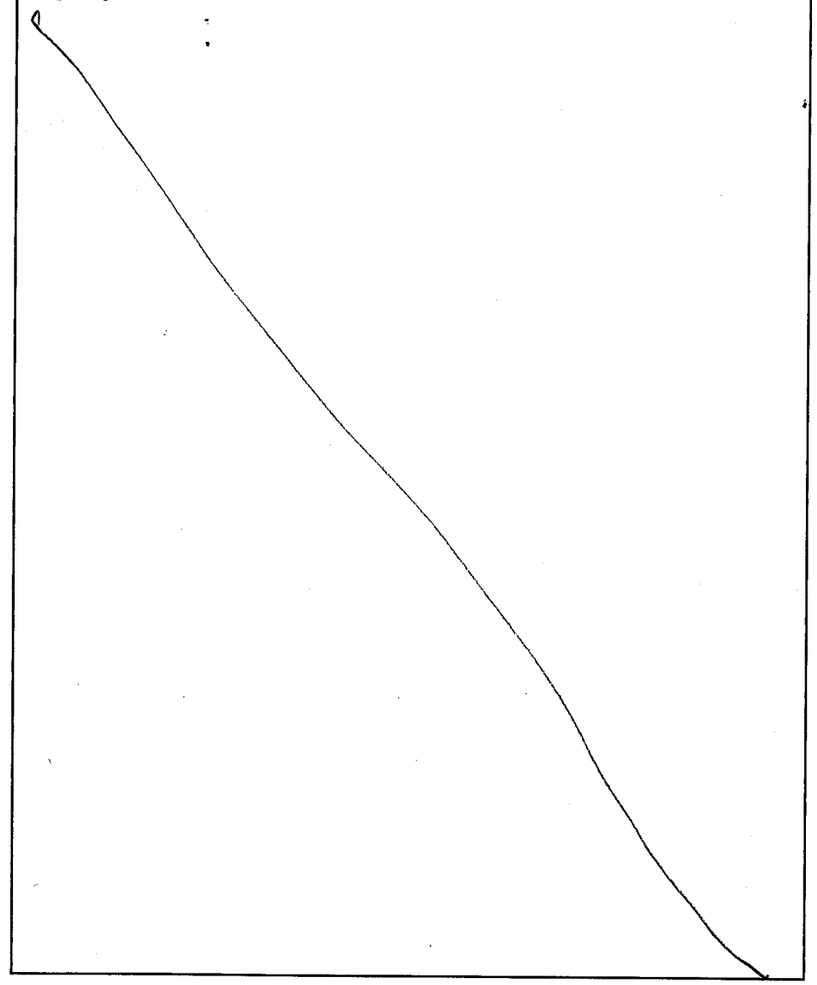
Jun 19 1998 10:19 **26May98.log** Page 1

File: viewNet.c++

Function: void introPopup(Widget parentw)

Purpose:

To make the splash screen a little more flashy and highlight the fact that it was developed by CNWRA. Ran into trouble with displaying pixmaps.



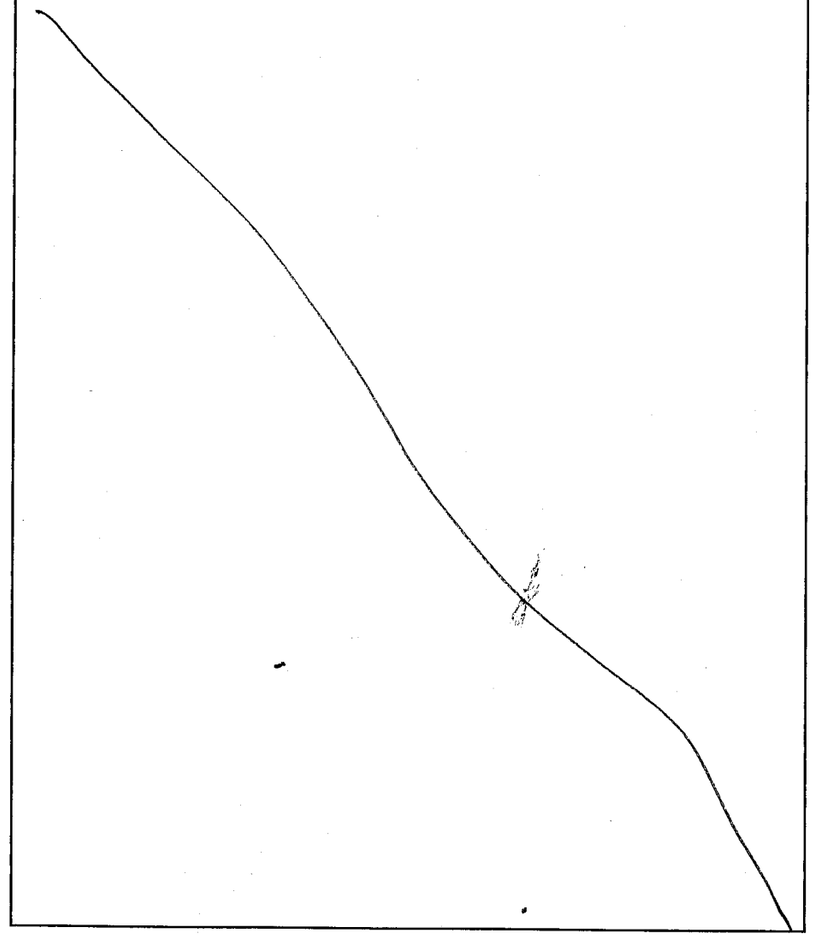
JWB

File: viewNet.c++

Functions: void introPopup(Widget parentw)
void introClose(Widget, XtPointer, XtPointer)

Modifictations:

Radically changed the splash screen. The graphics had to be added as one pixmap overlaid on a button without a callback. Also, had to make the top level of the splash screen a global widget and the close callback a XtDestroyWidget call.



J W B

Files modified:

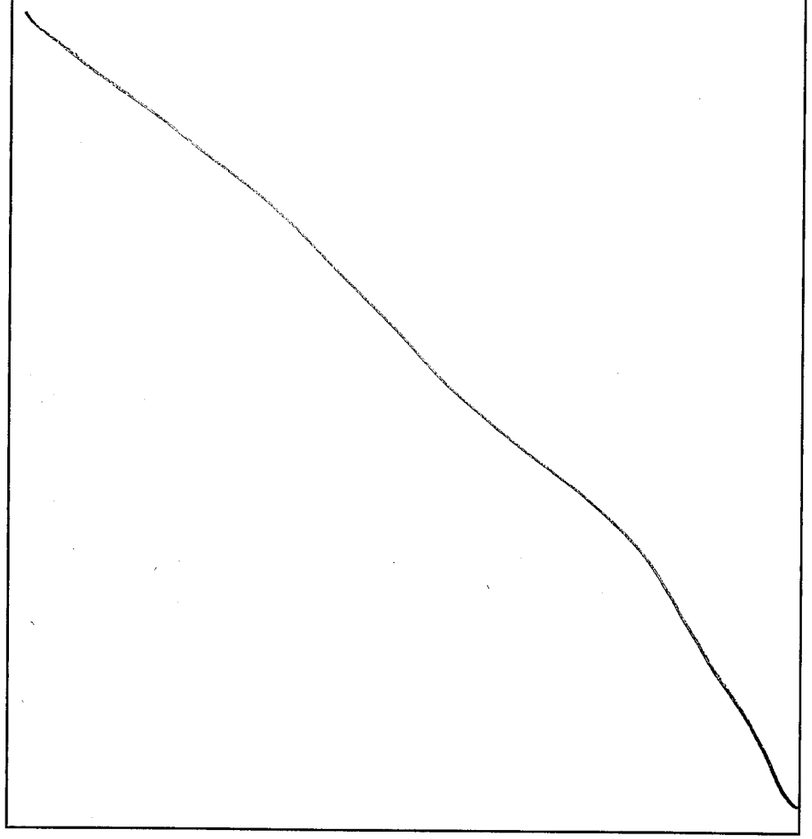
overlayWidgetClass.c++, overlayWidgetCB.hh,
and overlayWidgetCB.c++

Functions Affected and/or Added:

OverlayWidgetClass::initialize
void saveOvl(Widget inWidget, XtPointer, XtPointer)
void saveCBOvl(Widget w, XtPointer, XtPointer call_data)

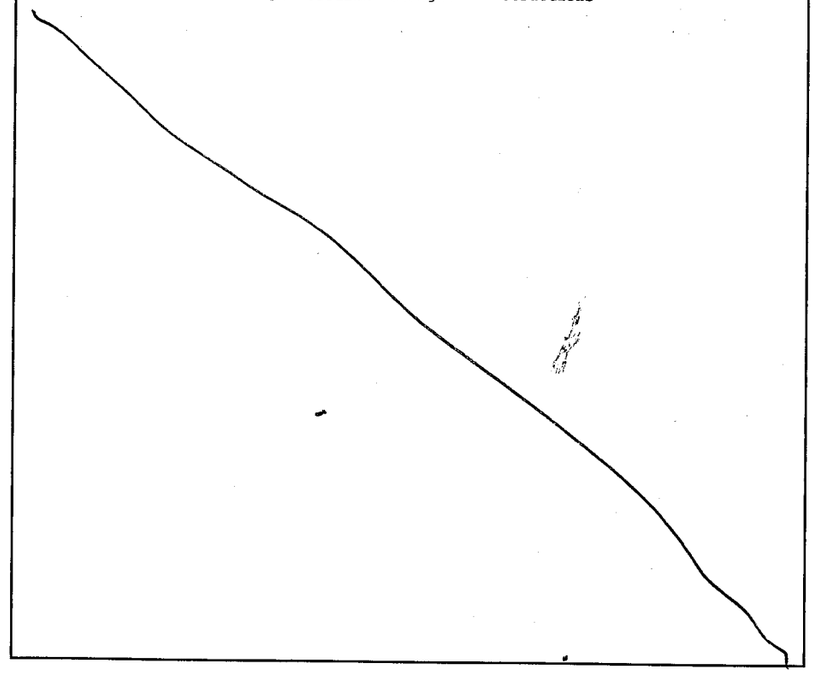
Purpose for Modifications:

Added superficial functionality and gui features to enable saving overlays in an application specific format designated by the file suffix '.3ds.ovr' The actual file write and load features have yet to be written. All the above will enable the user to save shape and color attributes for the overlays.



J W B

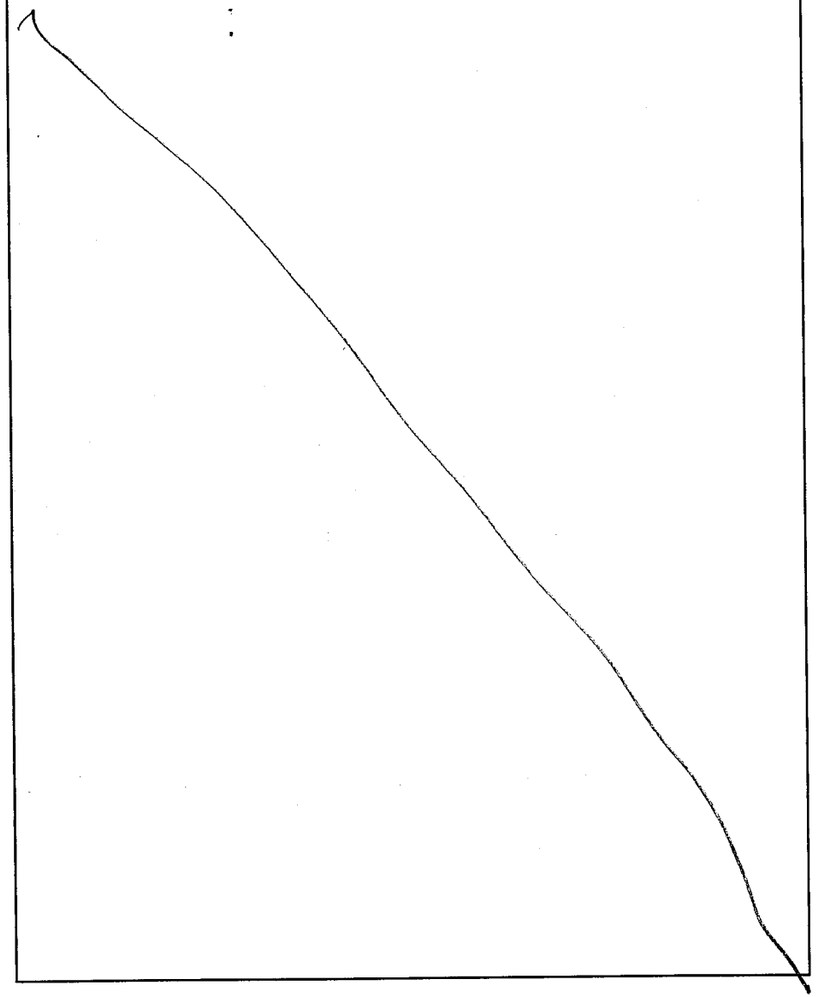
```
File:  cmdClass.c++

Changes:

in initialize and setTensileStr functions, reversed sign of tensile str
slider

File:  cmdClass_scaleCB.c++

Changes:

in scaleCB, switched sign so that tensile str representation
in calculations remains negative but is diplayed as positive.

File:  mohrCallbacks.c++

Changes:

in mohrCBdraw, changed displayed tensile str to positive

File: mohrOptionClass.c++

Changes:

in initialize, moved "Rock Info" button and renamed it "References"

==========

Revised user's manual as per Goodluck Ofoegbu's instructions
```



J W B

3June98.lo


```
pb = XtVaCreateManagedWidget("logo", xmPushButtonWidgetClass,
        frame, NULL);

XpmCreatePixmapFromData(dpy, DefaultRootWindow(dpy),
   stress_logo_xpm, &pix_returned, NULL, NULL);

XtVaSetValues(pb, XmNlabelType, XmPIXMAP,
   XmNlabelPixmap, pix_returned, NULL);
```



J W B

4June98.lo

File: mohrOptionClass.c++

Function: initialize

Added Code:

```
XtAddCallback (attributes.fluidText, XmNlosingFocusCallback,
               mohrChangeFluidField, NULL);
```

Purpose:

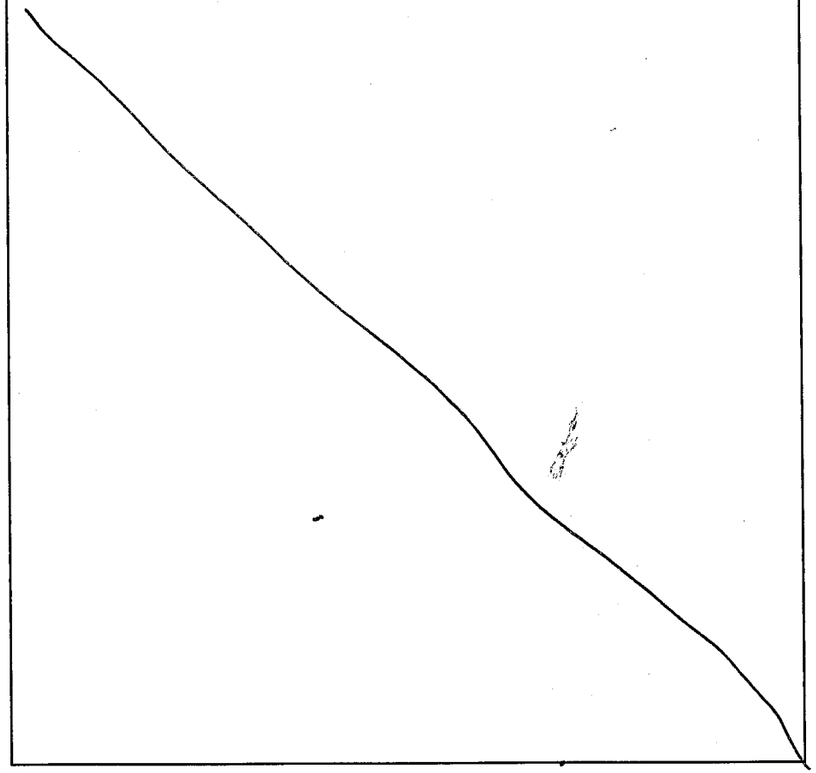
The actual value for fluid pressure will now be updated once the user changes the text field and moves on without pressing enter. The slider is also, of course, updated as well.

==========

File: viewNet.c++

Modification:

changed all references to 2.0 to 1.3 and removed alpha



JWB

File: viewNet.c++, stress.logo.xpm, Overview.sc

Purpose:

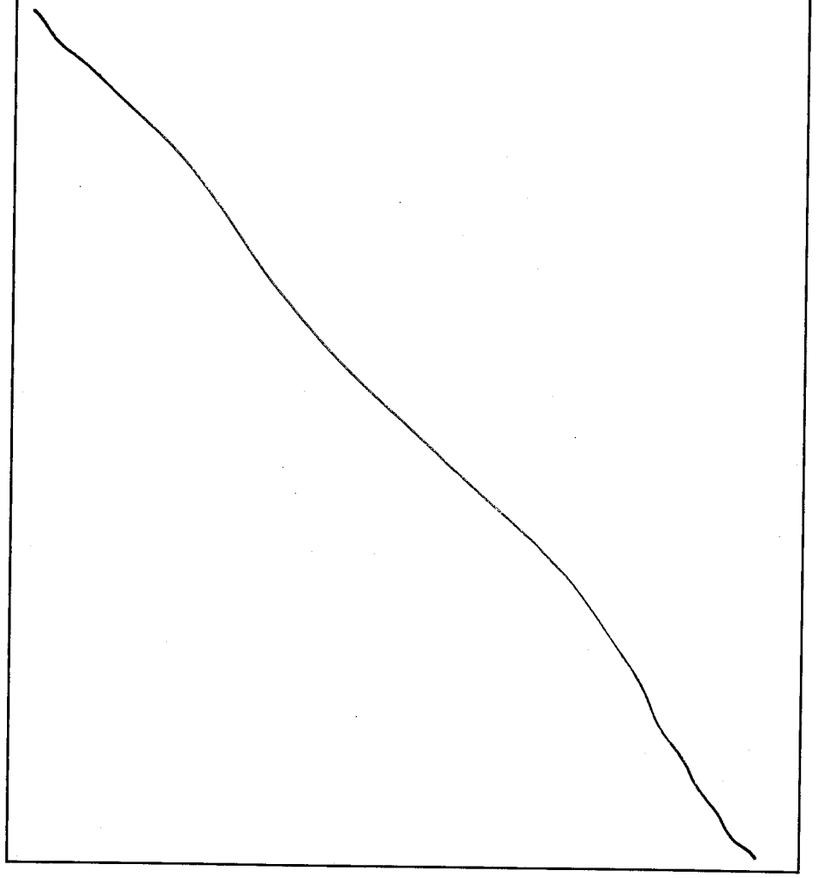
Added warnings about Beta release and replaced the splash screen logo with a better shadowed one

==========

File: mohrCallbacks.c++

Purpose:

Added tickmarks and tickmark labels to the Mohr graph



JWB

Aug 6 1998 08:47 **19June98.log** Page 1

File: cmdClass.hh

Modifications:

Made normalizations

Purpose for Modifications:

Fluid Pressure and Tensile Strength were not being normalized along with the stresses when Mohr settings are applied to the Slider and Tendency Plot.

Aug 6 1998 08:42 **16July98.log** Page 1

File Modified: Overview.sc

Modification Explanation:

Updated formulas for coefficient computation. They are now consistent with 3DStress operation.

==========

File Modified: plotClass_display.c++

Modifications:

Changed the formula displayed above the legend for Stress Ratio mode to read "tau over sigma sub n" instead of "sigma sub s over sigma sub n."

Aug 6 1998 08:42 **21July98.log** Page 1

Files Modified:

faultBuilderCB.c++
fileClass.c++
fileClass.hh
fileGlobals.hh
fileSystem.c++
fileSystem.hh
segmentClass.c++
segmentClass.hh
viewerCallbacks.c++

Reason for Modification:

The fault and line building tools were in a crazy, hacked state. This necessitated differentiating between 3D and 2D building in the code withing the segement and file classes. The builders should be pretty reliable now.

Aug 6 1998 08:42 **22July98.log** Page 1

Files Modified:

faultBuilderCB.c++
fileClass.c++
fileClass.hh
fileGlobals.hh
fileSystem.c++
fileSystem.hh
segmentClass.c++
segmentClass.hh
viewerCallbacks.c++
viewerClass.c++
viewerOptionCB.c++

Reason for Modification:

3D Fault Builder is now fully functional.

Aug 6 1998 08:42 **23July98.log** Page 1

Files Modified:

mapOptionCB.c++
mapOptionCB.hh
mapOptionClass.c++
mapOptionClass.hh

Explanation:

Added the ability to save 2D files and builder constructions to a .lin file.

File Modified: Appendix.sc

Explanation:

Added a warning about system requirments for producting .flt files via the run2grdToFlt script

---

Files Modified:

mohrOptionCB.hh
mohrOptionCB.c++

Explanation:

Added better error-trapping on the text fields.

Aug 6 1998 08:42 **24July98.log** Page 1

Files Modified:

lineBuilderCB.c++
mapButtonCB.c++
mapOptionCB.c++
mapOptionCB.hh
mapOptionClass.c++
mapOptionClass.hh
roseCallbacks.c++
viewNetGlobals.hh

Explanation:

Added the ability to load and remove multiple map files. The Rose diagram has yet to be altered to take care of multiple files though.

Files Modified:

lineBuilderCB.c++
lineBuilderCB.hh
lineBuilderClass.c++
lineBuilderClass.hh
faultBuilderCB.c++
faultBuilderCB.hh
faultBuilderClass.c++
faultBuilderClass.hh
fileSystem.c++
fileSystem.hh

Explanation:

Added ability to clear builder-made segments.

Aug 6 1998 08:42 **27July98.log** Page 1

Files Modified:

fileClass.c++,v
fileClass.hh,v
fileSystem.c++,v
fileSystem.hh,v
lineBuilderCB.c++,v
mapButtonCB.c++,v
mapOptionCB.c++,v
roseCallbacks.c++,v
segmentClass.c++,v
segmentClass.hh,v

Modification Explanation:

Rose diagram calculation procedures had to be modified to take multiple files and the line builder into account.

Files Modified:

lineBuilderCB.c++
mapButtonCB.c++
mapOptionCB.c++
roseClass.c++
roseClass.hh
roseOptionCB.c++
roseOptionCB.hh
roseOptionClass.c++
roseOptionClass.hh

Modification Explanation:

Added a button to toggle the inclusion of builder data into the rose diagram calculations.

JWB

14 August 1998

THIS NOTEBOOK DOCUMENTS PROGRAMMING WORK PERFORMED BY JOSH BUCKNER IN PREPARATION FOR RELEASE OF 3DSTRESS V. 1.3, WHICH WAS DELIVERED TO THE NRC ON 12 AUGUST 1998.

FUTURE 3DSTRESS PROGRAMMING WORK WILL BE DOCUMENTED IN A NEW SCIENTIFIC NOTEBOOK.

I HEREBY CLOSE THIS SCIENTIFIC NOTEBOOK.

J. P. Buckner

14 AUGUST 1998

I HAVE REVIEWED THIS SCIENTIFIC NOTEBOOK AND BELIEVE IT ADEQUATELY DOCUMENTS THE MODIFICATIONS TO 3DSTRESS SUCH THAT A PERSON WITH SIMILAR COMPUTER SKILLS AND EXPERIENCE COULD REDEVELOP THE MODIFICATIONS BY FOLLOWING ~~THIS~~ WHICH THE PROCEDURES OUTLINED IN THIS SCIENTIFIC NOTEBOOK

A. Lawrence McKeyne

8/14/98