



Normal mode visualization
by Madhusudhan Y Reddy

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in
Computer Science
Montana State University
© Copyright by Madhusudhan Y Reddy (1992)

Abstract:

The program consists of a Graphical User Interface (GUI) and related software for representing Normal Modes within the molecular structure.

The first stage of this project involves reading the molecular structure from the data files and displaying the molecular structure on the screen. The second stage involves representing the Normal Modes within the molecule using a color code.

The Graphical User Interface is built using X and Motif. The GUI consists of the following:

- o A ScrolledWindow which enables the user to scroll through a larger drawing area containing the molecular structure by sliding the ScrollBars.

- o A set of PushButtons which enables the user to perform functions such as: 1. selecting a data file; 2. plotting the molecular structure; 3. drawing the molecular bonds; 4. selecting the Normal Mode/Frequency to be displayed; 5. displaying the selected Normal Mode/Frequency, 6. displaying the chart for the color code.

We can also select a set of atoms by clicking on the screen and display the molecular vibrations corresponding to the most significant Normal Mode pertaining to that set of atoms.

- o A ScaleWidget which enables the user to scale the figure by moving the slider,
- o Three ScaleWidgets for rotating the figure around the three axes of rotation.

All the child widgets respond appropriately to resize requests from the parent widget.

NORMAL MODE VISUALIZATION

by

Madhusudhan Y. Reddy

A thesis submitted in partial fulfillment

of the requirements for the degree

of

Master of Science

in

Computer Science

Montana State University

Bozeman, Montana

March 1992

N378
R2475

APPROVAL

of a thesis submitted by
Madhusudhan Y. Reddy

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

March 18, 1992
Date

J. D. Stankley
Chairperson, Graduate Committee

Approved for the Major Department

March 18, 1992
Date

J. D. Stankley
Head, Major Department

Approved for the College of Graduate Studies

March 18, 1992
Date

Henry L. Parsons
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Dean of Libraries when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Matthew William Kelly

Date MARCH 18, 1992

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my graduate committee members *Dr. J. Denbigh Starkey*, *Mr. Ray Babcock* and *Dr. John Paxton* and the rest of the faculty members from the Department of Computer Science for their help and guidance during my graduate program.

I would also like to thank *Dr. Sam Helgerson* and *Dr. Edward Dratz* from the Department of Chemistry for their help and suggestions for my thesis.

v
TABLE OF CONTENTS

LIST OF FIGURES	vi
ABSTRACT	vii
INTRODUCTION	1
BUILDING THE GRAPHICAL USER INTERFACE	3
Scrolled Window	3
Scaling	5
Rotation	6
Push Buttons	7
Select Data File	8
draw_bonds	9
Select Normal_mode/Frequency	10
Show Nrm_modes	12
Draw fig.	12
Display Selection	13
Clear Selection	14
(Un)Display color code	15
Quit	15
VISUALIZING THE MOLECULAR STRUCTURE AND NORMAL	
MODES	16
Plotting the Molecular Structure	16
Color Code for Representing Vibrations	17
Displaying the Molecular Vibrations for a Given Normal Mode or Frequency	20
Displaying the Molecular Vibrations for a Given Set of Atoms	20
CONCLUSION	22
REFERENCES	23

LIST OF FIGURES

Figure		Page
1	Vibrational modes for the linear triatomic molecule carbon dioxide . .	2
2	Graphical User Interface	4
3	File Selection Box	9
4	Frequency Selection Box	11
5	Chart for calculating color intensities at different magnitudes of vibration	18

ABSTRACT

The program consists of a *Graphical User Interface (GUI)* and related software for representing Normal Modes within the molecular structure.

The first stage of this project involves reading the molecular structure from the data files and displaying the molecular structure on the screen. The second stage involves representing the Normal Modes within the molecule using a color code.

The *Graphical User Interface* is built using *X* and *Motif*. The GUI consists of the following:

- A *ScrolledWindow* which enables the user to scroll through a larger drawing area containing the molecular structure by sliding the ScrollBars.
- A set of *PushButtons* which enables the user to perform functions such as:
 1. selecting a data file;
 2. plotting the molecular structure;
 3. drawing the molecular bonds;
 4. selecting the *Normal Mode/Frequency* to be displayed;
 5. displaying the selected *Normal Mode/Frequency*;
 6. displaying the chart for the color code.

We can also select a set of atoms by clicking on the screen and display the molecular vibrations corresponding to the most significant Normal Mode pertaining to that set of atoms.

- A *ScaleWidget* which enables the user to scale the figure by moving the slider.
- Three *ScaleWidgets* for rotating the figure around the three axes of rotation.

All the child widgets respond appropriately to resize requests from the parent widget.

INTRODUCTION

Atoms within polyatomic molecules are in a constant state of motion due to the various interatomic and intermolecular forces that are active within. The most significant phenomenon causing this motion are the chemical bonds, which are caused by the sharing of electrons between atoms.

Most isolated molecules are characterized by having one equilibrium configuration [2, chapter 2]. There are some which have several equivalent equilibrium configurations with relative ease of interconversion among each. By a configuration, we refer to a three-dimensional arrangement of all nuclei.

Two types of vibration are possible within a single bond [1, chapter 3]. As seen in Figure 1, it may stretch along its axis, or it may bend. A simple analogy is that of two spheres joined by a weightless spring, and it is easy to see that less energy is required to bend the spring than to stretch it. Not surprisingly, it follows that the bending vibrations of chemical bonds require less energy and occur at lower frequencies than do the corresponding stretching modes. In polyatomic molecules there are many bonds and numerous vibrational opportunities exist. A structure containing n atoms, where $n > 2$, has in theory $3n - 6$ fundamental vibrational modes; of these, $n - 1$ are stretching oscillations and $2n - 5$ are bending motions.

A simple dynamic model of a polyatomic molecule neglects the coupling between vibration and rotation, and regards the motion as a combination of harmonic oscil-

lation and rigid rotation [2, chapter 2]. Such a model yields a reasonable description of the vibration-rotation bands for non-degenerate vibrations, but is spectacularly incorrect in accounting for the rotational structure of bands arising from degenerate vibrations. Hence, the importance of *Normal Modes* of vibration.

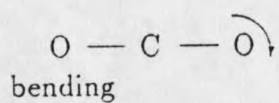
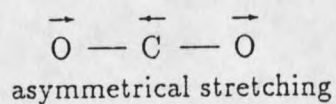
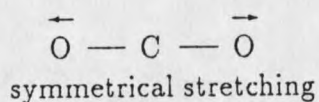


Figure 1: Vibrational modes for the linear triatomic molecule carbon dioxide

BUILDING THE GRAPHICAL USER INTERFACE

An important part of any application is the interface between the user and the application. For a graphics application, the *Graphical User Interface* (*GUI*) plays an important role in determining the success of the application. This section deals with the *GUI* of this application.

Scrolled Window

The GUI consists of a `ScrolledWindow` which enables us to scroll through a `DrawingArea`. The `ScrolledWindow` belongs to the `XmScrolledWindow` class of the motif widget set. It consists of two scroll bars, one vertical and one horizontal, and a viewing area which provides a visible window onto a larger data display, in this case the `DrawingArea`. The `XmNscrollingPolicy` resource of the `ScrolledWindow` is set to `XmAUTOMATIC`, in which case the `ScrolledWindow` automatically creates the `ScrollBars` and attaches callbacks to the `ScrollBars`. The `XmNscrollBarDisplayPolicy` resource of the `ScrolledWindow` is set to `XmSTATIC`, in which case the `ScrolledWindow` displays the `ScrollBars` whenever the `ScrolledWindow` is managed, regardless of the relationship between the view-area and the drawing-area. The `ScrolledWindow` resizes itself accordingly when the main window is resized.

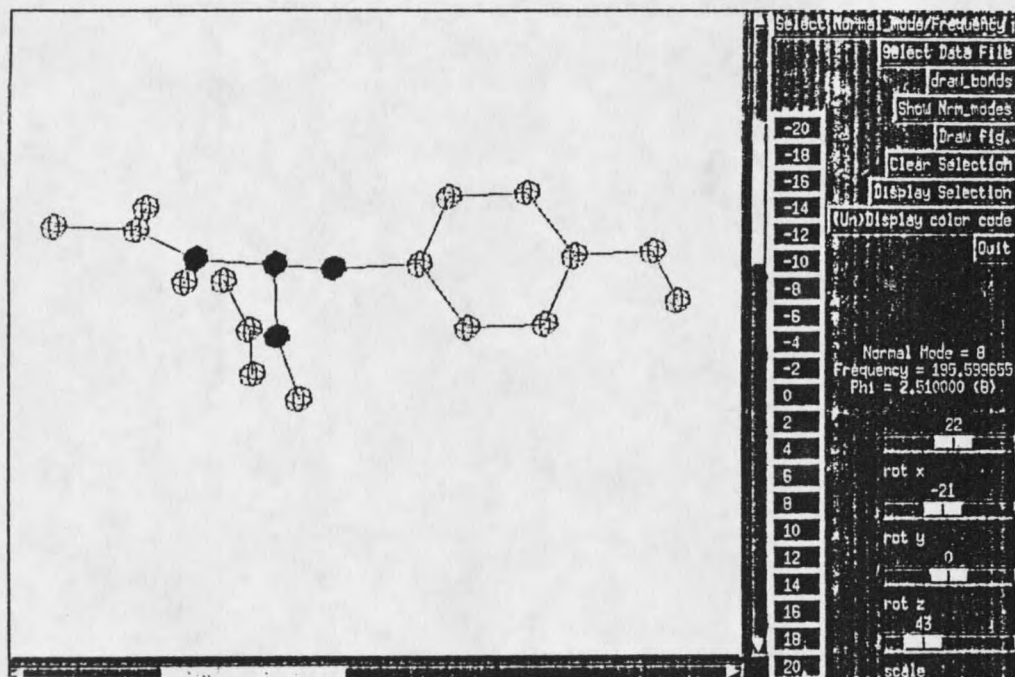


Figure 2: Graphical User Interface

The `DrawingArea` belongs to the `XmDrawingArea` class of the motif widget set, and it has a size of 2000×2000 pixels. The `XmNexpose` callback resource of the `DrawingArea` is set to the procedure `daCB()` only after the data file has been selected. The `DrawingArea` is the child of the `ScrolledWindow`.

The `XmNworkWindow` resource of the `ScrolledWindow` is set to `DrawingArea` which specifies that the `DrawingArea` is the work area that is to be scrolled by the `ScrolledWindow`. The `ScrolledWindow` is attached to the form widget on the top, left and bottom.

As can be seen in Figure 2, the `ScrolledWindow` has a horizontal scroll bar at the bottom and a vertical scroll bar to the right. By sliding the scroll bars, we can scroll through the drawing-area.

The source code for the `ScrolledWindow` can be found in the `main()` procedure

in the file `sb.c`.

Scaling

The `ScaleWidget` at the bottom right hand corner of the GUI enables us to scale the molecular structure. The scale factor can be set anywhere between 1 and 200. The `ScaleWidget` belongs to the `XmScale` class of the motif widget set. The `ScaleWidget` has an elongated rectangular region similar to a scroll bar. A slider inside this region indicates the current value along the scale. The user can modify the scale's value by moving the slider within the rectangular region of the scale.

The `XmNmaximum` resource of the `ScaleWidget` is set to 200 which limits the upper value of the `ScaleWidget` to 200. Similarly, the `XmNminimum` resource is set to 1 which limits the lower value to 1. The `XmNOrientation` resource is set to `XmHORIZONTAL` so that the `ScaleWidget` is positioned horizontally in the GUI. The `XmNprocessingDirection` resource is set to `XmMAX_ON_RIGHT` which means the scale value increases as we move the slider from left to right. The `XmNshowValue` resource is set to `TRUE` which creates a label and displays the current slider value next to the slider. The `XmNtitleString` resource is set to "scale" which creates a label and displays "scale" next to the `ScaleWidget`. The `XmNvalue` resource is set to 1 which sets the initial value of the `ScaleWidget` to 1. The `ScaleWidget` is attached to the `FormWidget` on the right and bottom. The source code for the `ScaleWidget` can be found in the procedure `create_scale()` in the file `scale.c`.

In order to scale the figure being displayed, the user can move the slider of the ScaleWidget and set the desired scale. Then, on redrawing the figure, the existing figure will be scaled according to the value set on the ScaleWidget by directly accessing the ScaleWidget to obtain its current value. The ScaleWidget's current value can be accessed with the motif system call *XmScaleGetValue()*. The scale factor is obtained while calculating the transformation matrix.

Rotation

The three scale widgets at the lower right hand side of the GUI enable the user to rotate the molecular structure about the three axes of rotation. The molecular structure can be rotated anywhere between -180° to $+180^\circ$, about any of the three axes x , y , or z . These widgets belong to the *XmScale* class of the motif widget set. These widgets are similar to the ScaleWidget described in the previous section. They consist of a rectangular region with a slider inside that indicates the current value. The user can select the angle of rotation about the three axes by moving the sliders of the respective widgets.

The following resources are set for the three widgets. The *XmNmaximum* resource is set to $+180$ which sets the upper limit of the angle of rotation to $+180^\circ$. Similarly, the *XmNminimum* resource is set to -180 which sets the lower limit of the angle of rotation to -180° . The *XmNorientation* resource is set to *XmHORIZONTAL* which positions the widgets horizontally in the GUI. The *XmNprocess-*

ingDirection resource is set to XmMAX_ON_RIGHT so that the value increases as we move the slider from left to right. The XmNshowValue resource is set to TRUE so that labels next to the widgets display the current slider values. The XmNtitleString resources of the three widgets are set to "rot x", "rot y" and "rot z", respectively, so that these labels are displayed next to the widgets. The XmNvalue resource is set to 0 so that the initial values of all three widgets is set to 0. These widgets are attached to the FormWidget on the right and to the ScaleWidget at the bottom. The source code for the creation of these widgets can be found in the procedure *create_rot_scales()* in the file *scale.c*.

In order to rotate the figure about any axis by any degree, the user can set the angles by moving the sliders of the respective widgets. On redrawing, the figure will be rotated about the three axes according to the current values of the three widgets. The values of the three widgets are accessed directly by using the motif system call *XmScaleGetValue()*. These values are obtained when the transformation matrix is being calculated.

Push Buttons

The GUI consists of nine PushButtons located at the top right hand side. These PushButtons belong to the XmPushButton class of the motif widget set. Each of these PushButtons has a callback associated with it. The actions performed by each of these buttons are as follows:

Select Data File

This button, on being pushed, calls the procedure *file_selct.buttonCB()*. This procedure creates and manages a FileSelectionDialog (FSD) which displays the list of data files. Figure 3 shows a FSD. The user can then select a data file by clicking the mouse button on the desired file name and then by clicking on the "OK" button of the FSD. The *XmNautoUnmanage* resource is set to FALSE which prevents the FSD from being automatically unmanaged when a button is selected. The *XmNmustMatch* resource is set to TRUE which prevents the user from entering an invalid file name in the text edit field. The *XmNselectionLabelString* resource is set to "Select a file" so that the label "Select a file" is displayed in the FSD. The *XmNdirMask* resource is set to "*.pdb" so that only files with the extension ".pdb" will be displayed. The *XmNnoResize* resource is set to TRUE so that the FSD cannot be resized by the user. The *XmNokCallback* and *XmNcancelCallback* resources are set to the procedure *fileCB()*. Also, the "HELP" button in the FSD is unmanaged.

The action taken by the procedure *fileCB()* depends on whether the user pushed the "OK" or "CANCEL" button in the FSD. If the "CANCEL" button is selected, this procedure simply unmanages the FSD, whereas, if the "OK" button is selected, this procedure extracts the file name into the global variable *f* and draws the entire molecular structure on the screen. Also, when a data file is selected for the first time, *XmNactivateCallback* resources are set for the other buttons and *XmNexpose-*

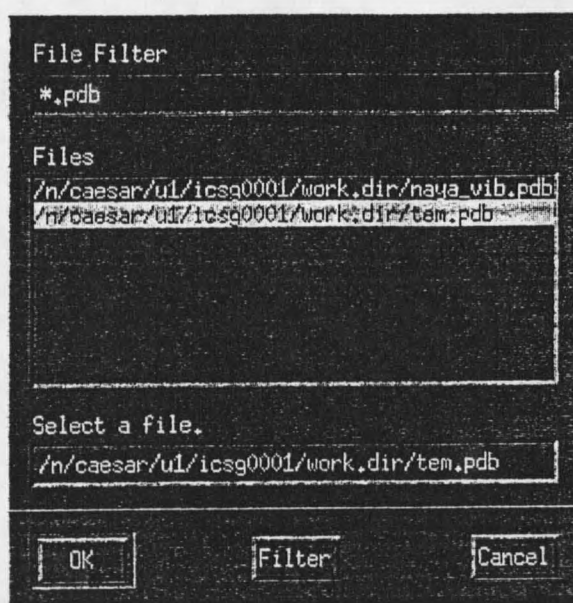


Figure 3: File Selection Box

Callback and XmNinputCallback resources are set for the DrawingArea.

The source code for all the procedures mentioned in this section appear in the file sb.c.

draw_bonds

This button calls the procedure *draw_bonds_buttonCB()*. This procedure first verifies if the filename contained in the global variable *f* is correct. If it is correct, it goes on to display the interatomic bonds in the molecular structure. This is accomplished by drawing the bonds between atoms according to a partly predetermined pattern. Each data file would require a different pattern and hence this procedure would have to be modified for every new data file. The source code for the procedure *draw_bonds_buttonCB()* can be found in the file bonds.c.

Select Normal_mode/Frequency

This button invokes the procedure *nrm_select_buttonCB()*. This procedure creates and manages a SelectionDialog which displays a list of normal modes along with the frequencies. The user can then select a normal mode or frequency by scrolling through the list and then pressing the “OK” button of the SelectionDialog. Figure 4 shows a SelectionDialog. The XmNautoUnmanage resource is set to FALSE so that the SelectionDialog is not automatically unmanaged when a button is pushed. The XmNselectionLabelString resource is set to “pick a frequency” so that the label “pick a frequency” is displayed in the SelectionDialog. The XmNnoResize resource is set to TRUE so that the SelectionDialog cannot be resized. The XmNokCallback and XmNcancelCallback resources are set to the procedure *dialogCB()*, and the HELP button is unmanaged.

The list of items to be displayed in the SelectionBox is generated by first invoking the procedure *read_nrm()*. This procedure reads the list of normal modes and frequencies from the data file into the arrays *nrm[]* and *freq[]* which are declared globally. This list is then assigned to the selection box by first converting them into an array *list[]* of type XmSTRING and then by setting the appropriate resources of the SelectionDialog. The XmNlistItems resource is set to *list[]* which specifies the list of items for the SelectionDialog, and the XmNlistItemCount resource is set to *list_cnt* which specifies the number of items in the SelectionDialog. Also, the XmNmustMatch resource is set to TRUE so that the user cannot select an invalid

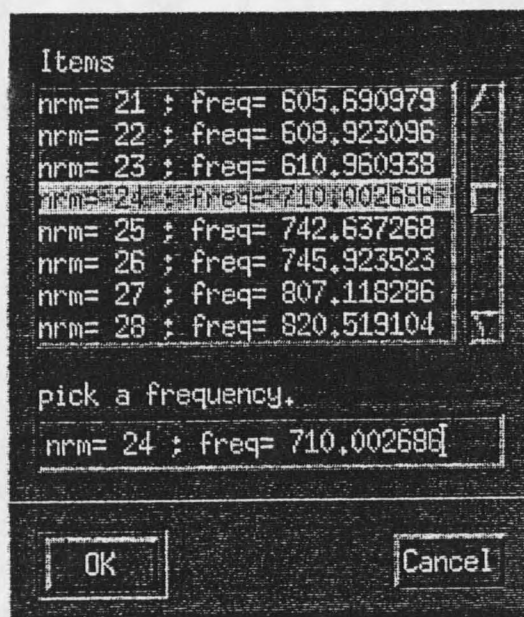


Figure 4: Frequency Selection Box

item.

When an item is selected from the SelectionDialog, the procedure *dialogCB()* is invoked. The action performed by *dialogCB()* depends on whether the “OK” or “CANCEL” buttons were selected in the SelectionDialog. If the “CANCEL” button was selected, the procedure simply unmanages the SelectionDialog. Whereas, if the “OK” button was selected, the values of the selected normal mode and frequency are retrieved into the global variable *s*. Then the procedure *read_phi()* is invoked. This procedure scans the data file to locate the most active part of the molecular structure corresponding to the selected normal mode and frequency. This value is stored in the global variable *code[]*.

The source code for the procedures *nrm_selct_buttonCB()* and *dialogCB()* can be found in the file *sb.c*. The source code for *read_nrm()* can be found in the file

`rd_nrm.c` and the source code for `read_phi()` can be found in the file `rd_phi.c`.

Show Nrm_modes

This button invokes the procedure `disp_nrm_mode_buttonCB()`. This procedure first sets up the foreground color according to the value stored in `code[0].phi` and then invokes one of the procedures `nrm1()` through `nrm20()`, depending on the value stored in `code[0].n`. The procedures `nrm1()` through `nrm20()` fill a set of atoms with the foreground color, setup by `setup_nrm_fg()`. The foreground color setup by `setup_nrm_fg()` is explained in Chapter 3.

The source code for the procedure `disp_nrm_mode_buttonCB()` can be found in the file `nrm_mode.c` and the source code for the procedure `setup_nrm_fg()` can be found in the file `gc.c`.

Draw fig.

This button invokes the procedure `draw_buttonCB()`. The procedure `draw_buttonCB()` first sets up the graphics context, clears the DrawingArea, and then redraws the entire molecular structure. The molecular structure is redrawn by invoking the procedure `do_drawing()`. The procedure `do_drawing()` first calculates the scale matrix by fetching the scale factor from the ScaleWidget. It then invokes the procedure `read_coord()` which reads the molecular coordinates into the global array `molecule[]`. The rotation matrix is then calculated by invoking the procedure `get_rot_mat()`. Fi-

nally, the transformation matrix is calculated by multiplying the scale matrix and the rotation matrix. The original coordinates of the molecular structure are multiplied by the transformation matrix and the resulting values are plotted on the screen. Also, the plotted values are stored in the array `plotted_pts[]`. Each atom is represented on the screen by three octagons in the three different planes. This is accomplished by invoking the procedure `draw_oct[]` thrice, each time passing a different axis.

The source code for the procedures `draw_buttonCB()` and `do_drawing()` is located in the file `draw.c`. The source code for the procedure `read_coord()` can be found in the file `rd_coord.c`. The source code for `get_rot_mat()` is in the file `rotation.c` and the source code for `draw_oct()` is in the file `octagons.c`.

Display Selection

This button, on being pressed, displays the magnitude of motion corresponding to the most significant Normal Mode for a given set of atoms. Before we can utilize this function, we have to select a set of atoms from the molecular structure on the screen. This can be done by placing the cursor over the atom and clicking the mouse button. On selecting an atom, it is blackened in order to indicate that the atom has been selected. The program lets us select up to a maximum of four atoms at a time.

The button "Display Selection" invokes the procedure `display_selection_bu-`

tttonCB(). This procedure in turn invokes *da_ip_nrm()*. This procedure checks to see if the combination of atoms is a valid one. If it is, then it invokes the procedure *read_cphi()* which scans the data file to retrieve the value of the magnitude of motion corresponding to the most significant Normal Mode for the given combination of atoms. This value is stored in the global variable *cphi*. *da_ip_nrm()* then invokes *setup_da_ip_fg()* which sets up the foreground color as explained in Chapter 3. Then, one of procedures *nrm1()* through *nrm20()* is invoked to display the Normal Mode. If the combination of atoms is invalid, a label "No Match" is displayed.

The source code for the procedure *display_selection_buttonCB()* is located in the file *ip_buttons.c*. The source code for *da_ip_nrm()* is in the file *da_ip.c* and the source code for *read_cphi()* is in the file *rd_da_ip_phi.c*.

Clear Selection

This button clears the set of atoms that were selected, as explained in the previous section, and enables the user to select a new set of atoms. This is done by invoking the procedure *clear_selection_buttonCB()*.

The source code for the procedure *clear_selection_buttonCB()* is in the file *ip_buttons.c*.

(Un)Display color code

This button alternately displays and removes a color chart that shows the range of colors used to represent the Normal Modes. This color chart consists of 21 labels, each with a different color. The topmost label is blue and the color changes to red gradually with each succeeding label. The generation of colors for representing Normal Modes is explained in Chapter 3. This is done by invoking the procedure *disp_color_buttonCB()*. This procedure, when called for the first time, creates the color chart by calling *disp_col_seq()* and thereafter alternately manages and unmanages the color chart, i.e. it behaves like a ToggleSwitch.

The source code for the procedure *disp_color_buttonCB()* is in the file *ip_buttons.c* and the source code for the procedure *disp_col_seq()* is in the file *chart.c*.

Quit

This button terminates the execution of the entire program. It invokes the procedure *quit_buttonCB()*.

The source code for the procedure *quit_buttonCB()* is in the file *ip_buttons.c*.

VISUALIZING THE MOLECULAR STRUCTURE AND NORMAL MODES

This section deals with plotting the molecular structure and displaying the normal modes according to the instructions given by the user.

Plotting the Molecular Structure

The molecular structure is plotted on the ScrolledWindow in response to three events. They are, when a data file is selected, when the window is scrolled and when the DrawingArea receives an exposure event. The XmNexposeCallback resource of the DrawingArea is set to the procedure *daCB()*. When the molecular structure needs to be drawn/redrawn, the procedure *do_drawing()* is invoked. This procedure reads in the molecular coordinates from the data file, calculates the transformation matrix after taking into consideration the scale factor and the rotation angles about the three axes and, finally, plots the molecular structure on the ScrolledWindow after performing the necessary transformations. The final values of the molecular structure that were plotted are stored in the array `plotted_pts[]` for reference by other procedures. The working of the procedure *do_drawing()* is explained in detail in Chapter 2.

The source code for the procedures *daCB()* and *do_drawing()* is located in the file `draw.c`.

Color Code for Representing Vibrations

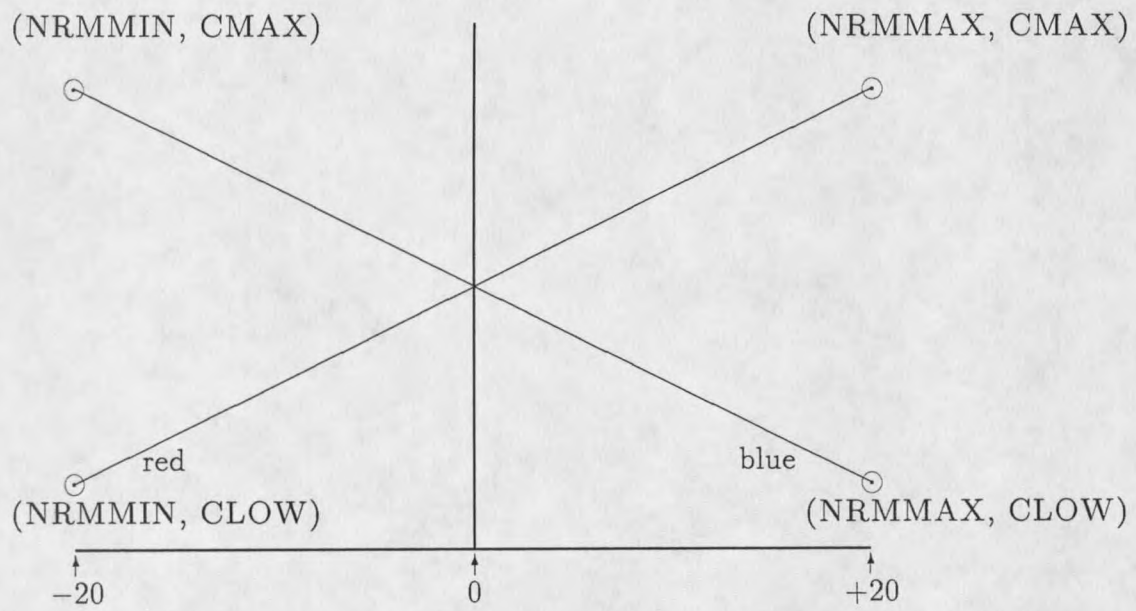
This application uses varying color shades to represent vibrations within the molecular structure. The color chart varies from a complete blue at the top to a complete red at the bottom of the chart. The transition from blue to red is gradual.

In X-windows colors are specified by the intensity of each of their red, green and blue (RGB) components [3, chapter 6]. The values of these components can range from 0 to 65535 where 0 corresponds to the lowest intensity of a color component and 65535 corresponds to the highest intensity. The X-server scales these values to the range of color intensities supported by the hardware. The Xlib function *XAllocColor()* allocates a read-only entry in a colormap. This function requires an *XColor* structure containing the RGB components of the color to be allocated.

In this application the intensity of blue gradually decreases from **CMAX** (65535) to **CLOW** (16380). Similarly, the intensity of red gradually increases from **CLOW** (16380) to **CMAX** (65535). The intensity of green is always **CMIN** (0). Figure 5 shows a graph of the RGB intensities used for the application. In order to calculate the pixel value for a particular intensity of vibration, we would first calculate the slopes of the lines representing the intensities of the colors red and blue.

The slope for red is given by :

$$rslope = \frac{CMAX - CLOW}{NRMMAX - NRMMIN}$$



CMAX	65535
CLOW	16380
NRMMAX	+20
NRMMIN	-20

Figure 5: Chart for calculating color intensities at different magnitudes of vibration

and the slope for blue is given by :

$$bslope = \frac{CMAX - CLOW}{NRMMIN - NRMMAX}$$

Once the slopes have been calculated, we can calculate the intensities for red and blue as follows :

The intensity (pixel value) of red is :

$$color.red = ((val - NRMMAX) * rslope) + CMAX;$$

and the intensity (pixel value) of blue is :

$$color.blue = ((val - NRMMIN) * bslope) + CMAX;$$

where *val* is the intensity of vibration. The value of *val* varies between NRMMIN (-20) to NRMMAX (+20).

The color chart that is displayed on pressing the button “(Un)Display color code” consists of 21 labels, each with a different color. The RGB components for these labels are calculated at uniform intervals using the above technique.

This method is used for generating different colors by the procedures *setup_nrm_fg()*, *setup_da_ip_fg()* and *disp_col_seq()*. The source code for *setup_nrm_fg()* and *setup_da_ip_fg()* is in the file *gc.c* and the source code for *disp_col_seq()* is in the file *chart.c*.

Displaying the Molecular Vibrations for a Given Normal Mode or Frequency

For a given normal_mode/frequency the program displays the most active part of the molecule. We can select any normal_mode/frequency by pressing the “Select Normal_mode/Frequency” button. This button pops up a SelectionDialog which enables the user to scroll through the list of normal modes and select one. Then, on pressing the button “Show Nrm_modes”, the program displays the intensity of vibrations for the most active part of the molecule. The intensity is represented using a color chart. The program scans the data file in order to locate the most active part of the molecule for the given normal mode. The information is then stored in a global variable where it is accessed by other procedures that display the intensity on the screen. The procedures that perform this task are *dialogCB()* and *read_phi()*. The function of the procedures is explained in detail in Chapter 2.

The source code for *dialogCB()* is in the file *sb.c* and the source code *read_phi()* is in the file *rd_phi.c*.

Displaying the Molecular Vibrations for a Given Set of Atoms

For a given set of atoms the program can display the intensity of vibrations corresponding to the most significant normal mode for that set of atoms. The user

can select an atom by placing the cursor over the atom and clicking the mouse button. In order to indicate that an atom has been selected, it is filled with a black foreground. The user can select up to a maximum of four atoms at a time. Then, on pressing the button "Display Selection", the intensity of vibrations corresponding to the most significant normal mode for that set of atoms is displayed. The intensity is represented using a color chart. The program scans the values in each normal mode in the data file to locate the most significant normal mode. The information is stored in a variable for use by other procedures. This task is performed by the procedures *da_ip_nrm()* and *read_cphi()*. The working of these procedures is explained in Chapter 2.

The source code for the procedure *da_ip_nrm()* is in the file *da_ip.c* and the source code for *read_cphi()* is in the file *rd_da_ip_phi.c*.

CONCLUSION

This program was completed successfully with the helpful guidance of the faculty members from the Departments of Computer Science and Chemistry. Although the program works quite satisfactorily, certain improvements can be made.

At present, the user has to redraw the figure after selecting a new scale or angle in order to be able to view the structure after the changes have been made. This can be improved so that the program automatically redraws the figure after a new scale or angle has been set, i.e. as we move the slider in the ScaleWidget or one of the rotation widgets, the program must be able to reflect the changes immediately on the screen.

Also, the program at present represents the magnitude of vibration with the help of plain colors. This can be improved by shading the atoms which will give a better appearance to the molecular structure.

Another area with a scope for improvement is that of representing the molecular motion itself. At present the program uses a method of coloring to indicate motion. This can be improved to show some form of motion on the screen.

REFERENCES

- [1] Brown, Floyd and Sainsbury. *Organic Spectroscopy*. John Wiley & Sons, New York, NY, 1988.
- [2] Decius and Hexter. *Molecular Vibrations in Crystals*. McGraw-Hill, New York, NY, 1977.
- [3] Douglas A. Young. *The X Window System : Programming and Applications with Xt - OSF/Motif edition*. Prentice-Hall, Englewood Cliffs, New Jersey, 1990.
- [4] Foley, Van Dam, Feiner, Hughes. *Computer Graphics : Principles and Practice. 2nd edition*. Addison-Wesley, Reading, Massachusetts, 1990.
- [5] Open Software Foundation. *OSF/Motif Programmer's Reference*. Prentice-Hall, Englewood Cliffs, New Jersey, 1990.
- [6] *The X Window System in a Nutshell*. Edited by Daniel Gilly and Tim O'Reilly. O'Reilly & Associates, Sebastapol, CA, 1990.

MONTANA STATE UNIVERSITY LIBRARIES



3 1762 10177553 2

HOUCHEN
BINDERY LTD
UTICA/OMAHA
NE.