The Water-Wave Signal Analysis

Graduate Project

By

Lei Lu

May, 1998

Committee Members

Dr. David Thomas,
Chairman

Dr. Mario Guimaraes,
Member

Dr. Holly Patterson,
Member
<table>
<thead>
<tr>
<th>Table of Contents</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>Background &amp; Rational</td>
<td>4</td>
</tr>
<tr>
<td>Narrative</td>
<td>6</td>
</tr>
<tr>
<td>Project Environment</td>
<td>11</td>
</tr>
<tr>
<td>Project Procedures</td>
<td>12</td>
</tr>
<tr>
<td>End Results</td>
<td>26</td>
</tr>
<tr>
<td>References</td>
<td>29</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
</tr>
</tbody>
</table>
Abstract

This project develops a windows-based system. It allows the user to use the Fast Fourier Transform (FFT) to convert discrete digital signals/values (measured in time and elevation) into a series of sine and cosine functions based on the frequency content of the signal. These functions are then presented graphically.
Background and Rational

This report is submitted in partial fulfillment of requirements for a master's degree in Computer Information Science at Texas A&M University-Corpus Christi.

This project will analyze water wave information from an offshore monitoring location. By applying this project to the different monitoring stations, a thorough understanding of the water waves can be obtained and analyzed for use in scientific research.

Fourier Transform Theory has existed for years and is effectively used to analyze discrete periodical signals (reference [1]. Page 10-19). Water waves can be categorized into periodical signals and non-periodical signals. Periodical signals are produced by motion of the Earth, the Sun, the Moon, the depth of the water, and other regular factors. Non-periodical signals are produced by those factors such as storms, earthquakes, heavy tankers passing by monitoring locations, and so on. Periodical influences are predictable. Non-periodical influences are not predictable and have been causing disasters and economical damages for long time. This project uses Fourier Transform Theory to analyze water waves by way of wave frequencies, amplitudes, phases and how they are distributed in the frequency domain. It can also find a series of very low frequency (reference [2]) patterns. This project can be extended by applying those additional irregular influences to water waves at specific times, with specific amounts of energy, based on existing information. Moreover, it could be used to predict the water elevation in a certain area near the monitoring stations and could estimate if it is possible that a flood disaster situation is present.

This project extracts the M lowest frequencies (refer to later contents of this chapter) existing in the signal domain and analyzes them graphically. Darker gray values emphasize those frequencies with higher amplitudes, and lighter gray values indicate those frequencies with smaller amplitudes in a diagram. It has menu options for user to choose if data needs to be filtered in a certain way. An image file and a hard copy of the results can be provided.

The Fast Fourier Transform used in this project is called Cooley Turkey FFT (reference [1], 2. 2.9 page 48). When Radix 2 Cooley Turkey is used, it
computes the first $M$ frequencies where $M$ complies with $2^M=N$ ($N$ is the number of total samples) (reference [3]).
Narrative

The major parts of this project are: Data Processing, Data Presentation, and On Line Help.

1. Data Processing

Processing data involves filtering the original data, windowing the data and computing amplitude and initial phase for each frequency of each data frame, and then saving this information in a file for the data presentation process.

- **Data Filtering:**
  The purpose of data filtering is to reduce the influence of high frequencies. There are many ways to filter out high frequencies, such as: several point average, polynomial fit, and so on (reference [3]). However, over modifying the signals will cause a loss of frequencies, while under modifying the signals will only partially cut off the undesirable high frequency influence. It requires a testing process to smooth out the signal variations and to sufficiently cut off high frequency influence at the desired the cut-off frequency. Therefore, a proper way to smooth the data is the key to properly reduce the unwanted high frequencies. In order to have this program accommodate different purposes, multiple choices of data filtering methods will be provided for the user to choose from. The user is also required to input the size of the frame with the number of the signals in the given time domain. The larger the frame size is, the more frequencies there can be. For example, with FFT method for the frame size of 1024, ten frequencies can be obtained. Nine frequencies can be obtained from frame size of 512. And eight frequencies can be obtained from a frame size of 256. Also, seven frequencies can be obtained from frame size of 128 signals (Frame sizes less than 128 will not be considered).

- **Windowing Data:**
  Normally, it is required to have a set of data changing at least one complete cycle in the time domain. The real data does not behave this way. A modification is needed to force the data to have at least
one complete cycle. This adjustment can be done in numerous ways. The most common way is to take 10\% of the data on each side of the frame and use a cosine function to bring the data to 0. Under these circumstances, the data in each data frame will have at least one complete cycle, and a reasonable result can be reached with Fast Fourier Transform method (reference [3]). This process requires discrete signals and a chosen size for the frame. The output of the process are windowed or modified discrete signals with a cosine function. It is unclear, if the windowing data process is appropriate. From the software developer's point of view, it is left for the user as an option to choose between using windowing data or not.

- **FFT Data:**
  This uses Fast Fourier Transform (FFT) to compute each frequency, amplitude and its initial phase in the selected data domain (512, 1024 or 4096 as pre-selected)

2. **Data Presentation**
   Data presentation involves graphical representation of the raw data (such as elevation and time). The cosine curve of each frequency with its amplitude and initial phase will be displayed. The amplitude for each frequency will be indicated by the pixel's gray value. Since this project requires a few graphical presentation, JAVA is used to accomplish whole programming task. Only one window frame is used performing this project. This frame includes several displaying windows and menus, such as a raw-data displaying window, a frequency-displaying window, a strength-displaying window for amplitudes, and a menu of choices for presetting.

- **Raw Data Displaying Window:**
  This window will use a polygon to join the elevations sequentially one after another. It only includes one frame while the time frame is being processed. As time moves forward, the second-frame data-set information will be used to update the previous frame. This graph will be changing like a moving picture. A Raw-data display window depicts the behavior of raw data within the given domain. It includes all of the data frames, but only displays them one-at-a-time, and one-after-another.
• Frequency Displaying Window:
  This window displays the frequencies of several cosine functions. These curves will be shown in different colors to differentiate them from each other. It too displays one frame at-a-time and one frame after-another-frame.

• Amplitude Strength Displaying Window
  This window displays computed amplitudes of each frequency in a frame with a vertical bar. On this bar the darker area (with higher gray values) shows the frequencies with higher amplitudes, and lighter areas (with lower gray values) represent the frequencies with lower amplitudes. And, the highest frequency is on the top, and the lowest frequency is at the bottom. If there are a total of n data frames, there will be n such vertical bars, and they are lined up one-after-another as time progresses. In this way, a graph is formed.

3. Online Help
  "Online Help" offers "HOW-TO" service and guides the user through from the beginning to the end.
The Water-Wave Signal Analysis
A Step by Step Operation of the System As Seen by the User

1) User Input
   a Data Filtering or None
   b Windowing Data or None
   c Size of Data Frame
   d Input Data File & Path
   e Output Image File Name
   
   All the choices must be made
   The Input file will be verified
   The Output file does not exist

2) Raw Data Window
   A graph shows a polygon of water elevation vs time
   graph 1

3) Frequency Display Window
   A graph shows the sine/cosines of each frequency with different colors
   graph 2

4) Amplitude Strength Display Window
   A vertical bar for each data frame from lowest to highest frequencies
   Darkness of the vertical bar indicates the strength of amplitude
   graph 3

5) Online Help
   Track the most recent input
   Output solution for panic

When user provides inputs correctly, 2), 3), 4) will display and 5) prompts the "Next To Do"
1. The user will be prompted with "WHAT TO DO" every time he/she choose the option. The user is suggested to follow the next instruction unless he/she fully understands procedures.

2. The user needs to choose Input File.

3. This option chooses the frame size ranging from 256, 512, 1024, 2048, and 4096.

4. The user chooses whether to use data filtering or not. The user chooses whether to use windowing data or not.

5. The user chooses the amplitude scale.

6. The user chooses the water elevation scale.

7. The user chooses the frame scale.

8. The user chooses adjust gray value scale.

9. The user chooses the update speed.

10. The user chooses the submit.

*Purpose of adjusting the scale is to exaggerate the graphic presentation. It does not change the nature of the original contents of the data.
Project Environment

This windows-based software can be running under a compatible IBM/PC-computer with following features.

1. Windows 95 is required to run this program.
2. Sun Java 1.1 or higher is the platform of this program.
3. Windows 95 will run on a 486 PC or better with sufficient RAM and disk space.
4. Super VGA(1024*768) or similar is necessary to have clear image.
Project Procedures

The Water-Wave Signal Analysis Top Level Flow Chart

The project is designed to use object Oriented programming (OOP) methodology. A class called "grad" contains attributes and methods this project needs, and a class called "MenuOpt" provides the menu options for use to choose. Methods can be M1, M2, M3, M4, and M5. The attributes in use at any particular time are based on choices made by user. When a process starts running, the first message will be sent to get the user's choices. The second message will be sent to get an input file. Then we request the object to check output information. After the object completes its preparation process, the object starts M1, M2, M3, and M4 processes. M5 is an online help process. During preparation process, it provides "WHAT TO DO NEXT". While running processes from M1 to M4, it provides the information of what method the object is executing.
The First Page of The Project:

Graduation Project
Lei Lu
3/1998

Committee Chairman: David Thomas, Ph.D
Committee Member: Mario Guimaraes, Ph.D
Committee Member: Holly Patterson, Ph.D

User Mouse Input:

This is a instance of MenuOpt class. It provides the menu input. By mouse clicking on the menu the data input can be executed.

They are Files, Frame Size, Data Filtering, Window Data, Amplitude Scale, Water Elevation Scale, Frame Scale, Adj. Gray Value, Adj. updateSpd, and Submit.
File:

It can input the data file by clicking appropriate file name.

Frame Size:

User can choose frame size by clicking a appropriate menu item.

Data Filtering:

User can choose Data Filtering option by clicking a menu item.
Window Data:

User can choose Data Filtering option by clicking a menu item.

Amplitude Scale:

User can choose appropriate amplitude scale adjusting to appropriate area.

Water Elevation Scale:

User can choose appropriate scale adjusting the water elevation to appropriate area.
**Frame Scale:**

User can adjust width of the graph displaying on the screen.

**Adjust Gray Value:**

This is used to adjust initial gray value of the power strength on the screen to an acceptable appearance.

This is used to adjust scale of the power strength on the screen to an accept appearance.
Adjust Update Speed:

User can adjust screen update whenever a close look needed to delay next screen update.

Submit:

User can execute the program, or exit the program.
Input Data File:

<table>
<thead>
<tr>
<th>Sample of The Input File</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997 05 10 00 36 32 625 952.79</td>
</tr>
<tr>
<td>1997 05 10 00 36 32 750 946.54</td>
</tr>
<tr>
<td>1997 05 10 00 36 33 875 959.04</td>
</tr>
<tr>
<td>1997 05 10 00 36 34 000 948.00</td>
</tr>
<tr>
<td>1997 05 10 00 36 35 000 984.86</td>
</tr>
<tr>
<td>1997 05 10 00 36 35 250 1020.05</td>
</tr>
<tr>
<td>1997 05 10 00 36 35 375 1029.01</td>
</tr>
<tr>
<td>1997 05 10 00 36 36 125 939.67</td>
</tr>
<tr>
<td>1997 05 10 00 36 36 250 914.68</td>
</tr>
</tbody>
</table>

Record Format For Input File

<table>
<thead>
<tr>
<th>Year</th>
<th>Month</th>
<th>Day</th>
<th>Hour</th>
<th>Minute</th>
<th>Second</th>
<th>Second/1000</th>
<th>Height (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>05</td>
<td>10</td>
<td>00</td>
<td>36</td>
<td>32</td>
<td>750</td>
<td>946.54</td>
</tr>
</tbody>
</table>

Purpose:

It provides the water elevation and its time to M1 and M5.

Input:

Time and water elevation as formatted above.

Output:

It passes time and water elevation to M1 and M5.
Purpose:

Interpolation process fills in the missing data with linear interpolation.

Input:

The input data is the input data file which are in above format, and the output data are time and corresponding water elevation. It can be described like this mathematically, (Ti, Ei) where 0 < i < "Signal" in a frame. After this interpolation process, the input data set is modified in the same time interval, if there is any data was missing. It is then passed to M2.

Output:

The interpolated data without missing data.

Mathematical Model:

For a moment Ti, there is a pair of time and elevation values corresponding to (Ti, Ei), and also (Tj, Ej) for moment j. Assuming there is data missing k, and k is such that Ti < Tk < Tj, we have the following equation to compute the elevation corresponding to that Tk.

\[ E_k = E_i + \frac{(T_k - T_i)}{(E_i - E_j)} \times \frac{(E_i - E_j)}{(T_i - T_j)} \]
1. Purpose of Data Filtering:

N-point average means that for a moment of Ti, there is a water elevation Ei. Instead of using Ei in the computation process, we use an averaged Ei for an average of N values (N is normally from 3 or 5) around that moment.

Input:

It takes interpolated data.

Output:

It outputs filtered data.

The Mathematical model is:

\[ E_i = \sum (E_j), \]
where \( i - \text{int}(N/2) < j < i + \text{int}(N/2). \)

For the data at the beginning and ending of the frame, this process can be omitted.

2. Purpose of Windowing Data:

This process brings 10% of the data values on each side of frame to zero based on a cosine function.

Input:

It takes filtered data or interpolated data (when the option is not chosen).

Output:

It outputs windowed data.

The mathematical model is:

For the beginning side of data in the frame:

\[ E_{\text{windowed}} = E_i \times \cosine \{ (i/(0.1 \times N)) \times 90 \}, \]

Or
E_{\text{windowed}} = E_i \times \cos \left\{ \frac{i}{N} \times 90 \right\}.

Where \( N \) is the number of the data values in the frame and \( i \) is the sequential order of the data values in the frame and its range is \( 0 \leq i \leq 0.1 \times N \).

For the ending side of data values in the frame:

\[ E_{\text{windowed}} = E_i \times \cos \left\{ \left[ 1 - \left( \frac{N-i}{0.1 \times N} \right) \right] \times 90 \right\}, \]

Where \( N \) is the number of the data values in the frame and \( i \) is the sequential order of the data values in the frame and its range is \( 0.9 \times N \leq i \leq N \).

3. Purpose of FFT Data:

It decomposes water waves into frequencies with Fourier Transform.

Input:

It takes windowed data or un-windowed data (when the option is not chosen).

Output:

It outputs \( M \) amplitudes and initial phases.

Cooley Tukey Radix-2 FFT program is adopted from reference [1] at page 110 in FORTRAN, and rewrites in JAVA.

(This project is focused on the data presentation graphically, the mathematical model is not part of discussion in this project. For the reference of FFT, please refer to related topics in reference [1].)
This process involves the screen coordinate system. Which is increasing from left to right and from top to bottom.

Purpose:

This process uses the results to generate graphs.

Input:

Interpolated data.

Output:

Graphs of water elevation.
1. Raw-Data Displaying Window:

   In the coordinate system of this window, the coordinate system is
designed from bottom to top as the water elevation increases and from
left to right as the time increases. A sequence of line segments connects
the \((T_i, E_i)\) \((0 < i < n)\) sequentially from the starting moment to the
ending moment.

2. Frequency Displaying Window:

   In the coordinate system of this window, the vertical axis from bottom
up indicates increasing amplitude, and horizontal axis increasing from
left to right indicates the increasing of time in that data frame.

   After completing Fast Fourier Transformation, \(M\) pairs of amplitude
and initial phase are obtained, where \(M\) complies with the equation.

   \[2^M = N.\]

<table>
<thead>
<tr>
<th>Relation between M and N</th>
</tr>
</thead>
<tbody>
<tr>
<td>M=</td>
</tr>
<tr>
<td>N=</td>
</tr>
</tbody>
</table>

   There will be total of \(M\) frequencies for each frame of the total \(N\) data
values.

3. Amplitude Strength Displaying Window:

   In the coordinate system of this window, the vertical axis increasing
from bottom up indicates increasing values from \(M=1\), to \(M=\text{last } M\),
and the horizontal axis increasing from left to right indicates one-frame-
after-another sequentially. The darker areas indicate the larger
amplitude, and the lighter areas indicate the smaller amplitude.
M4: THE WINDOW PRINTING & COPYING

Purpose:

The purpose of Window Printing and Copying is to have a softcopy and a hardcopy.

Input:

The screen image is the input.

Output:

The output prints a hardcopy a printer and writes a softcopy on the disk.
M5: THE ONLINE HELP

INPUT DATA FILE
TIME & WATER HEIGHT
PASSING EVENTS INFO TO M5

USER KEYBOARD INPUT:2
OUTPUT FILE NAME
PASSING EVENTS INFO TO M5

USER KEYBOARD INPUT:1
DATA FILTERING OR NONE
WINDOWING DATA OR NONE
SIZE OF DATA FRAME
PASSING EVENTS INFO TO M5

M5: Online Help

TRACKING EVERY EVENTS
RESPONSE "WHAT TO DO NEXT"

Purpose:

It provides a "WHAT TO DO NEXT". In case, the user loses track of what to do, the user can refer to the information from the help module.

Input:

M5 is tracking every event from "INPUT DATA FILE", "USER KEYBOARD INPUT:1", and "USER KEYBOARD INPUT:2".

Output:

It prompts helpful hints for user when he/she is in a panic.
The End Results

Complete documentation includes:

Testing samples:

1. Another sample test data generated from this function.

\[ F(t) = 40\cos(t) + 20\cos(2t+30) + 10\cos(4t+60) + 750. \]
2. A sample of real data from 5/10/97 10hr 36 min 32 sec to 23 hr 39 min 9 sec
Conclusion:

The results of the testing data and real data indicate that this is a successful approach to analyze the low frequencies of the water waves with FFT method.

From the testing data and real data a series of curves indicate how the activity of the low frequencies composed in the water waves. The frequency graph indicates the powers of each frequency visually, and the frequency strength graph shows the relations between the data frames.

The program itself provides flexibility during run time. A few parameters can be changed to emphasis the data presentation such as Data Filtering, Window Data, Amplitude Scale, Water Elevation Scale, Frame Scale, Adjust Gray Value, and Update Speed.
References


3. Dr. Robert Benson:
   Unpublished conversation concerning data filtering, etc.


7. Web Site http://math.math.sunysb.edu/~tony/tides/


Appendix

Main Program

1. This is class of grad.
import java.applet.Applet;
import java.awt.*;
import java.io.*;
import java.io.*;

public final class grad extends MenuOpt{
    private String buf=new String();
    private long Curr=0,Prev=0,StartTime=0;
    private double WaterElev,PrevWaterElev,CurrWaterElev;
    private double pt1=0,pt2=0;
    private int Yr,Day,Min,Sec,SecDec,M,TotSig=0,FrameSequence=0;
    private int FrameSize,FrameNumber=0,AmpStrNum=0;
    private int Time []=new int[7];

    private double AmpStr [][];
    private double Water [];
    private double Amplitude [];
    private double WatElev [];
    private double WatComp [];
    private double WR [];
    private double WI [];
    private double Phase [];
    private double Amp [];

    private boolean DISPLAY_READY=false;

    void DynA(int n){
        FrameSize=n;
        M=(int)(Math.log((double)FrameSize)/Math.log((double)2.0)+.01);
        WatElev=new double[n+1];
        Water =new double[n+1];
        WatComp=new double[n+1];
        WR =new double[n+1];
    }
g.drawLine((int)(FrameScale*(double)(j-1))+50,110-
(int)(WaterScale*(Water[j]-Amp[1])),
(int)(FrameScale*(double)j)+50,110-
(int)(WaterScale*(Water[j+1]-Amp[1])));
}
g.setColor(Color.black);
g.drawLine((int)(FrameScale*(double)j)+50,50,
(int)(FrameScale*(double)j)+50,270);
g.drawString("Frame Ends",(int)(FrameScale*(double)(j-1))+53,105);
//Draw Frequencies in Sine Function with their amplitudes.
for(int k=2;k<=M;k++){
    pt1=Amp[k]*Math.cos(Phase[k]*Math.PI/180.0);
g.setColor(setC(k));
    for(j=0;j<FrameSize;j++){
        pt2=Amp[k]*Math.cos(2.0*Math.PI*(double)(k-1)*(double)(j+1)/(double)FrameSize+
        Phase[k]*Math.PI/180.0);
        g.drawLine(50+(int)(FrameScale*(double)j),220-(int)(pt1*AmpScale),
        50+(int)(FrameScale*(double)(j+1)),220-(int)(pt2*AmpScale));
        pt1=pt2;
    }
}
//draw amplitude strength graph

g.setColor(Color.black);
g.drawString("Frequency Strength Graph",25,300);
g.drawString("# of Cycles",15,330);

//put current amp in the graph

for(int k=2;k<=M;k++){
    int tmp3=320+k*15;
    AmpStr[FrameNumber][k-2]=Amp[k];
g.drawString(String.valueOf(k-1),35,tmp3);
    if(k==M){
        g.drawString("Frame Number ->",30,tmp3+40);
    }
}
g.drawString("Amplitude=>",170,320);
for(int jj=0;jj<256;jj=jj+5){
    Color color=new Color(255-jj,255-jj,255-jj);
    g.setColor(color);
    g.fillRect(190+jj,287,5,15);
    if(jj%25==0){
        int tmpValue=(int)((jj-MidGray)/(int)GrayScale);
        if(tmpValue>=0 & & tmpValue<256){
            g.setColor(Color.black);
            tmp1=193+jj;
            g.drawString(String.valueOf(tmpValue),tmp1,320);
            g.drawLine(187+jj,307,190+jj,302);
            g.drawLine(193+jj,307,190+jj,302);
            g.drawLine(190+jj,307,190+jj,320);
        }
    }
}
g.drawString("Frame Size="+(int)(FrameSize/8)+" sec.",tmp1+30,310);

for(int f=0;f<=FrameNumber;f++){
    int UpperLeftX=0,UpperLeftY=0;
    for(int k=2;k<=M;k++){
        int temp1=(int)(AmpStr[f][k-2]*GrayScale+MidGray);
        temp1=(temp1>255)? 255:temp1;
        temp1=255-temp1;
        Color color=new Color(temp1,temp1,temp1);
        g.setColor(color);
        UpperLeftX=50+f*5;UpperLeftY=275+(k+2)*15;
        g.fillRect(UpperLeftX,UpperLeftY,5,15);
    }
    if(f%10==0){
        g.setColor(Color.black);
        g.drawLine(UpperLeftX,UpperLeftY+20,UpperLeftX,UpperLeftY+35);
        g.drawLine(UpperLeftX-3,UpperLeftY+20,UpperLeftX,UpperLeftY+15);
        g.drawLine(UpperLeftX+3,UpperLeftY+20,UpperLeftX,UpperLeftY+15);
    }
}
g.drawString(String.valueOf((140*((FrameSequence-1)/140)+f),UpperLeftX+5,UpperLeftY+30);
}
tmp1=UpperLeftX;
}
g.setColor(Color.black);
for(int jj=2;jj<=M;jj++){
g.drawString("f="+(double)(jj-1)/(double)(FrameSize/8)+" (cycle/sec)",tmp1+15,345+(jj-2)*15);
}
FrameNumber++;
if(FrameNumber==140){
    FrameNumber=0;
    AmpStrNum=0;
    DISPLAY READY=false;
    while(DISPLAY READY=false){
}
}
for(long ii=0;ii<SndScale;ii++){
    repaint();
}

private Color setC(int k){
    int clr;
    clr=k%3;
    if(clr==0){return Color.red;}
    else if(clr==1){return Color.blue;}
    else{return Color.black;}
}

void InitValue(){
    double tmp=2.0/(double)FrameSize*Math.PI;
    // System.out.println("M="+M+"*");
    // for(int i=1;i<=FrameSize;i++){WatElev[i]=0.0;WatComp[i]=0.0;}
    for(int i=1;i<=FrameSize/2;i=i+1){
        double a=(double)(i-1)*tmp;
        WR[i]=Math.cos(a);WI[i]=Math.sin(a);
    }
}
void FFT()
{
    int N1,N2,K,L,IE,IA;
    double E,tmp;
    N2=FrameSize;
    while(N2/2>0){
        N1=N2;N2=N2/2;IE=FrameSize/N1;IA=1;
        for(int j=1;j<=N2;j++){
            double C=WR[IA],S=WI[IA];
            IA=IA+IE;
            for(int i=j;i<=FrameSize;i=(i+N1)){
                L=i+N2;
                double XT=WatElev[i]-WatElev[L];
                WatElev[i]=WatElev[i]+WatElev[L];
                double YT=WatComp[i]-WatComp[L];
                WatComp[i]=WatComp[i]+WatComp[L];
                WatElev[L]=XT*C-YT*S;
                WatComp[L]=XT*S+YT*C;
            }
        }
    }
    //sequence the frequencies
    //Reverse counter action
    int k,j=1;N1=FrameSize-1;
    for(int i=1;i<=N1;i++){
        if(i<j){
            double XT=WatElev[j];WatElev[j]=WatElev[i];WatElev[i]=XT;
            XT=WatComp[j];WatComp[j]=WatComp[i];WatComp[i]=XT;
        }
        k=FrameSize/2;
        while(k<j) {
            j=j-k;k=k/2;
        }
        j=j+k;
    }
}
//computation of initial phase and amplitude
for(int i=1;i<=FrameSize;i++){
    if(i<=M){
        Amp[i]=Math.sqrt(WatComp[i]*WatComp[i]+WatElev[i]*WatElev[i]);
        if(i==1){Amp[i]=Amp[i]/(double)FrameSize;}
        else{Amp[i]=2.0*Amp[i]/(double)FrameSize;}
        if(WatElev[i]==0.0){
            Phase[i]=90.0;
        }
    } else {
        Phase[i]=Math.atan(WatComp[i]/WatElev[i])*180.0/Math.PI;
        if(WatElev[i]<0.0){
            Phase[i]=Phase[i]+180.0;
        }
        else{
            if(WatComp[i]<0.0){
                Phase[i]=Phase[i]+360.0;
            }
        }
    }
}

//Output:
System.out.println("Amp["+i+"]="+Amp[i]+",Phase["+i+"]="+Phase[i]);
WatComp[i]=0;WatElev[i]=0;
}

void ReadInputFile(String inputfile){
    FileInputStream waterElevation=null;
    try{
        waterElevation=new FileInputStream(inputfile);
    } catch(Exception e) {
        System.out.println(e);
        System.exit(1);
    }
    DataInputStream WH=new DataInputStream(waterElevation);
    TotSig=1;
while(true){
    try{buf=WH.readLine();}
    catch(IOException e){System.out.println(e);System.exit(1);}
    if(buf==null){break;}
    ToInt();
    if(TotSig-1==FrameSize){
        //do data filtering
        if(DataFiltering==true){
            AverageN(5);
        }
    }
    //do windowing data
    if(WindowData==true){
        WindowDataStart();
        WindowDataEnd();
    }
    for(int j=0;j<=FrameSize;j++){Water[j]=WatElev[j];}
    while(DISPLAY_READY==true){}
    FFT();
    FrameSequence();
    DISPLAY_READY=true;
    TotSig=1;
}
}

void FrameSequence(){
    help.appendText("This is the "+(FrameSequence+1)+"th Frame\n");
    FrameSequence++;
}
void ToInt(){
    int i,j=0,k=0;
    String tmp1=new String();
    String tmp=new String();
    for(i=0;i<buf.length();i++){
        if(buf.charAt(i)=='){
            Time[j]=Integer.valueOf(buf.substring(k,i)).intValue();
            Time[6]=(int)(((float)Time[6])/125+.05)*125;
            k=i+1;j++;
        }
    }
}
WaterElev=Double.valueOf(buf.substring(k,i)).doubleValue(); 
CurrWaterElev=WaterElev;

if(StartTime==0){
    StartTime=Curr;Prev=Curr;
}

// 1 sec. is the allowance for missing data
if(Curr-Prev>1000){StartTime=0;Curr=0;Prev=0;TotSig=1;return;}
while(Curr-Prev>130){interpolation();}
Prev=Curr;
PrevWaterElev=CurrWaterElev;
WatElev[TotSig]=CurrWaterElev;
TotSig=TotSig+1;

void interpolation(){
    long dtime,time1,time0;
    double dElev,intWaterElev;
    dtime=Curr-Prev;
    dElev=CurrWaterElev-PrevWaterElev;
    intWaterElev=PrevWaterElev+dElev/((double)dtime)*125;
    Prev=Prev+125;
    PrevWaterElev=intWaterElev;
    WatElev[TotSig]=intWaterElev;
    if(TotSig==FrameSize){return;}
    TotSig++;
}

void WindowDataEnd(){
    double Edge=(int)(0.1*(double)FrameSize),Edge1=(int)(0.9*(double)FrameSize)+1;
    for(int i=FrameSize;i>Edge1;i--){
        double degInt=(90.0*((double)i-Edge1)/Edge)*Math.PI/180.0;
        WatElev[i]=WatElev[i]*Math.cos(degInt);
    }
}
void WindowDataStart()
{
    int Edge=(int)(0.1*FrameSize);
    //System.out.println("Edge="+Edge);
    for(int i=1;i<Edge;i++){
        double degInt=(90.0-90.0*(double)(i-1)/(double)Edge)*Math.PI/180.0;
        WatElev[i]=WatElev[i]*Math.cos(degInt);
    }
}

void AverageN(int n){
    //for n can only choose 3,5. It can work when n=7, data lose its character.
    for(int i=1;i<=FrameSize;i++){
        if(i>=(n/2+1) & & i<FrameSize-(n/2+1)){
            double tmp=0.0;
            for(int j=i-n/2;j<=i+n/2;j++){
                tmp=tmp+WatElev[j]/n;
            }
            WatElev[i]=tmp;
        }
    }
    // System.out.println("tmp="+tmp);
}

void gradRun()
{
    help.appendText("Choose Input Data File\n");
    setFont(new Font("grad",Font.BOLD+Font.ITALIC,24));
    while(Submit()!=true){}
    this.Dyna(FrameSize());
    this.InitValue();
    setFont(new Font("grad",Font.PLAIN,12));
    this.ReadInputFile(InFile());
}

void LastInfo()
{
    help.appendText("Program Completes\n");
}

public static void main(String args[]){
    grad g=new grad();
}
g.gradRun();
// LastInfo();
//System.exit(0);

2. This is class of MenuOpt.

import java.awt.*;

public class MenuOpt extends Frame {

    protected Frame grad = new Frame("Home Comming Project");
    protected FileDialog openFileDialog=new FileDialog(grad,"Open Data File",FileDialog.LOAD);
    protected FileDialog saveDialog=new FileDialog(grad,"Save Data File",FileDialog.SAVE);

    protected String inFile = new String();

    protected boolean inAnApplet = true;
    protected boolean submit=false;
    protected boolean windowdata=false;
    protected boolean datafiltering=false;
    protected static boolean Pause_Dis=true;

    protected int   FrameSize=256;
    protected long  SpdScale=10000;
    protected double AmpScale=1.0,WaterScale=1.0,FrameScale=1.0,GrayScale=1.0,MidGray=100.0;

    protected TextArea help;

    public void paint(Graphics g){
        g.drawString("Testing",50,100);
    }
}
public MenuOpt() {

    super("Home Coming Project");
    inFile=null;
    //outFile=null;
    help= new TextArea(4,50);
    help.setEditable(false);
    add("South",help);

    //Build the menu bar.
    MenuBar menuBar = new MenuBar();
    setMenuBar(menuBar);

    //the 0th Menu:Files
    Menu Files= new Menu("Files", true);
    menuBar.add(Files);
    MenuItem InputFile = new MenuItem("Input File");
    Files.add(InputFile);

    //the first Menu:Frame Size
    Menu FrameSize=new Menu("Frame Size",true);
    menuBar.add(FrameSize);
    MenuItem Frame256=new MenuItem("Frame Size:256");
    MenuItem Frame512=new MenuItem("Frame Size:512");
    MenuItem Frame1024=new MenuItem("Frame Size:1024");
    MenuItem Frame2048=new MenuItem("Frame Size:2048");
    MenuItem Frame4096=new MenuItem("Frame Size:4096");
    FrameSize.add(Frame256);
    FrameSize.add(Frame512);
    FrameSize.add(Frame1024);
    FrameSize.add(Frame2048);
    FrameSize.add(Frame4096);

    //Second Menu:Data Filtering
Menu DataFilter = new Menu("Data Filtering", true);
menuBar.add(DataFilter);
MenuItem DataFileterYes = new MenuItem("Data Fileter:Yes");
MenuItem DataFileterNo = new MenuItem("Data Fileter:No");

DataFilter.add(DataFileterYes);
DataFilter.addSeparator();
DataFilter.add(DataFileterNo);

//Third Menu: Window Data

Menu WindowData = new Menu("Window Data", true);
menuBar.add(WindowData);
MenuItem WindowDataYes = new MenuItem("Window Data:Yes");
MenuItem WindowDataNo = new MenuItem("Window Data:No");
WindowData.add(WindowDataYes);
WindowData.addSeparator();
WindowData.add(WindowDataNo);

//Fourth Menu: Amplitude Scale

Menu AmplitudeScale = new Menu("Amplitude Scale", true);
menuBar.add(AmplitudeScale);
MenuItem F_reduce10 = new MenuItem("Reduce 10 times");
MenuItem F_reduce7 = new MenuItem("Reduce 7 times");
MenuItem F_reduce5 = new MenuItem("Reduce 5 times");
MenuItem F_reduce3 = new MenuItem("Reduce 3 times");
MenuItem F_reduce2 = new MenuItem("Reduce 2 times");
MenuItem F_same = new MenuItem("No change");
MenuItem F_increase2 = new MenuItem("Increase 2 times");
MenuItem F_increase3 = new MenuItem("Increase 3 times");
MenuItem F_increase5 = new MenuItem("Increase 5 times");
MenuItem F_increase7 = new MenuItem("Increase 7 times");
MenuItem F_increase10 = new MenuItem("Increase 10 times");

AmplitudeScale.add(F_reduce10);
AmplitudeScale.add(F_reduce7);
AmplitudeScale.add(F_reduce5);
AmplitudeScale.add(F_reduce3);
AmplitudeScale.add(F_reduce2);
    AmplitudeScale.addSeparator();
AmplitudeScale.add(F_same);
    AmplitudeScale.addSeparator();
AmplitudeScale.add(F_increase2);
AmplitudeScale.add(F_increase3);
AmplitudeScale.add(F_increase5);
AmplitudeScale.add(F_increase7);
AmplitudeScale.add(F_increase10);

//The Fifth Menu: Water Elevation Scale

Menu WaterScale=new Menu("Water Elevation Scale",true);
menuBar.add(WaterScale);
MenuItem W_reduce10 = new MenuItem("Reduce 10 times");
MenuItem W_reduce7 = new MenuItem("Reduce 7 times");
MenuItem W_reduce5 = new MenuItem("Reduce 5 times");
MenuItem W_reduce3 = new MenuItem("Reduce 3 times");
MenuItem W_reduce2 = new MenuItem("Reduce 2 times");
MenuItem W_same = new MenuItem("No change");
MenuItem W_increase2 = new MenuItem("Increase 2 times");
MenuItem W_increase3 = new MenuItem("Increase 3 times");
MenuItem W_increase5 = new MenuItem("Increase 5 times");
MenuItem W_increase7 = new MenuItem("Increase 7 times");
MenuItem W_increase10 = new MenuItem("Increase 10 times");

//The Sixth Menu: Frame Scale

WaterScale.add(W_reduce10);
WaterScale.add(W_reduce7);
WaterScale.add(W_reduce5);
WaterScale.add(W_reduce3);
WaterScale.add(W_reduce2);
    WaterScale.addSeparator();
WaterScale.add(W_same);
    WaterScale.addSeparator();
WaterScale.add(W_increase2);

43
WaterScale.add(W_increase3);
WaterScale.add(W_increase5);
WaterScale.add(W_increase7);
WaterScale.add(W_increase10);

Menu FrameScale=new Menu("Frame Scale",true);
menuBar.add(FramenScale);
MenuItem Frame_reduce10 = new MenuItem("Frame Reduce 10 times");
MenuItem Frame_reduce8 = new MenuItem("Frame Reduce 8 times");
MenuItem Frame_reduce5 = new MenuItem("Frame Reduce 5 times");
MenuItem Frame_reduce3 = new MenuItem("Frame Reduce 3 times");
MenuItem Frame_reduce2 = new MenuItem("Frame Reduce 2 times");
MenuItem Frame_same = new MenuItem("Frame scale no change");
MenuItem Frame_increase2 = new MenuItem("Frame Increase 2 times");

FrameScale.add(Frame_reduce10);
FrameScale.add(Frame_reduce8);
FrameScale.add(Frame_reduce5);
FrameScale.add(Frame_reduce3);
FrameScale.add(Frame_reduce2);
FrameScale.add(Frame_same);
FrameScale.add(Frame_increase2);

//Menu:Strength Scale:Gray Value Scale and Middle value
Menu adjGrayValue= new Menu("Adj. Gray Value",true);
menuBar.add(adjGrayValue);
Menu midValue=new Menu("Mid Gray Value");
MenuItem M75 =new MenuItem("Mid Gray Value=75");
MenuItem M100 =new MenuItem("Mid Gray Value=100");
MenuItem M125 =new MenuItem("Mid Gray Value=125");
MenuItem M150 =new MenuItem("Mid Gray Value=150");
MenuItem M175 =new MenuItem("Mid Gray Value=175");
midValue.add(M75);
midValue.add(M100);
midValue.add(M125);
midValue.add(M150);
midValue.add(M175);
adjGrayValue.add(midValue);
Menu GrayScale= new Menu("Adj. Scale",true);
adjGrayValue.add(GrayScale);
MenuItem S01 =new MenuItem("Gray Scale=0.1");
MenuItem S1 =new MenuItem("Gray Scale=1");
MenuItem S2 =new MenuItem("Gray Scale=2");
MenuItem S5 =new MenuItem("Gray Scale=5");
MenuItem S7 =new MenuItem("Gray Scale=7");
MenuItem S10 =new MenuItem("Gray Scale=10");
GrayScale.add(S01);
GrayScale.add(S1);
GrayScale.add(S2);
GrayScale.add(S5);
GrayScale.add(S7);
GrayScale.add(S10);
adjGrayValue.add(GrayScale);

//Menu:Strength Scale:Gray Value Scale and Middle value
Menu AdjSpd= new Menu("Adj. Update Spd",true);
menuBar.add(AdjSpd);
Menu Inc= new Menu("Increase");
MenuItem IncMore =new MenuItem("Increase More");
MenuItem IncLess =new MenuItem("Increase Less");
Inc.add(IncMore);
Inc.add(IncLess);
AdjSpd.add(Inc);

Menu Dec= new Menu("Decrease",true);
AdjSpd.add(Dec);
MenuItem DecMore =new MenuItem("Decrease More");
MenuItem DecLess =new MenuItem("Decrease Less");
Dec.add(DecMore);
Dec.add(DecLess);
AdjSpd.add(Dec);

//Menu:Submit and exit
Menu Submit= new Menu("Submit", true);
menuBar.add(Submit);
MenuItem execute=new MenuItem("Execute");
Submit.add(execute);
Submit.addSeparator();
MenuItem exit=new MenuItem("Exit");
Submit.add(exit);

//Menu completes

    resize(800,600);
    show();
}

public boolean handleEvent(Event event) {
    if(event.id == Event.WINDOW_DESTROY) {
        if(inAnApplet) {
            dispose();
        } else {
            System.exit(0);
        }
    }
    return super.handleEvent(event);
}

public boolean action(Event event, Object arg) {
    String str = "Action detected";
    String tmp1= null;
    if(event.target instanceof MenuItem) {
        MenuItem mi=(MenuItem)(event.target);
        str += " on " + arg;
        if(mi instanceof CheckboxMenuItem) {
            str+=" (state is "+(CheckboxMenuItem)mi).getState()+ ")";
        }
        MenuContainer parent = mi.getParent();
        if(parent instanceof Menu) {
            tmp1= ((Menu)parent).getLabel();
            str += " in " + ((Menu)parent).getLabel();
        } else {
            
}
str += " in a container that isn't a Menu";
}

// Menu response
if(arg=="Exit"){
    System.exit(0);
} else if(arg=="Execute"){
    if(inFile!=null){
        submit=true;
        help.appendText("It is time to run\n");
    } else {
        help.appendText("Input File is not ready\n");
    }
} else if(arg=="Increase More"){
    SpdScale=SpdScale/100;
    help.appendText("Increase Graph Speed More\n");
    help.appendText("Choose Submit\n");
} else if(arg=="Increase Less"){
    SpdScale=SpdScale/10;
    help.appendText("Increase Graph Speed Less\n");
    help.appendText("Choose Submit\n");
} else if(arg=="Decrease More"){
    SpdScale=SpdScale*100;
    SpdScale=(SpdScale<10)? 10:SpdScale;
    help.appendText("Decrease Graph Speed More\n");
    help.appendText("Choose Submit\n");
} else if(arg=="Decrease Less"){
    SpdScale=SpdScale*10;
    SpdScale=(SpdScale<10)? 10:SpdScale;
    help.appendText("Decrease Graph Speed Less\n");
    help.appendText("Choose Submit\n");
} else if(arg=="Gray Scale=0.1"){
    GrayScale=0.1;
    help.appendText("Gray Scale set=0.1\n");
    help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Gray Scale=1"){
    GrayScale=1.0;
    help.appendText("Gray Scale set=1\n");
help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Gray Scale=2"){
  GrayScale=2.0;
  help.appendText("Gray Scale set=2\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Gray Scale=5"){
  GrayScale=5.0;
  help.appendText("Gray Scale set=5\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Gray Scale=7"){
  GrayScale=7.0;
  help.appendText("Gray Scale set=7\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Gray Scale=10"){
  GrayScale=10.0;
  help.appendText("Gray Scale set=10\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Mid Gray Value=75"){
  //Mid gray value
  MidGray=75;
  help.appendText("Mid Gray set=75\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Mid Gray Value=100"){
  //Mid gray value
  MidGray=100;
  help.appendText("Mid Gray set=100\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Mid Gray Value=125"){
  //Mid gray value
  MidGray=125;
  help.appendText("Mid Gray set=125\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Mid Gray Value=150"){
  //Mid gray value
  MidGray=150;
  help.appendText("Mid Gray set=150\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Mid Gray Value=175"){
  MidGray=175;
  help.appendText("Mid Gray set=175\n");
  help.appendText("Choose Adjust Update Speed\n");
} else if(arg=="Frame Size:256"){
  //Frame Size

FrameSize=256;
    help.appendText("Frame Size:256\n");
    help.appendText("Choose Data Filtering\n");
} else if(arg=="Frame Size:512"){
    FrameSize=512;
    help.appendText("Frame Size:512\n");
    help.appendText("Choose Data Filtering\n");
} else if(arg=="Frame Size:1024"){
    FrameSize=1024;
    help.appendText("Frame Size:1024\n");
    help.appendText("Choose Data Filtering\n");
} else if(arg=="Frame Size:2048"){
    FrameSize=2048;
    help.appendText("Frame Size:2048\n");
    help.appendText("Choose Data Filtering\n");
} else if(arg=="Frame Size:4096"){
    FrameSize=4096;
    help.appendText("Frame Size:4096\n");
    help.appendText("Choose Data Filtering\n");
} else if(arg=="Input File"){
    //Files
    openDialog.show();
    inFile=openDialog.getFile();
    help.appendText("Input File="+inFile+"\n");
    help.appendText("Choose Frame Size\n");
} else if(arg=="Data Fileter:Yes"){
    //Data Filter
datafiltering=true;
    help.appendText(arg+"\n");
    help.appendText("Choose Window Data\n");
} else if(arg=="Data Fileter:No"){
    datafiltering=false;
    help.appendText(arg+"\n");
    help.appendText("Choose Window Data\n");
} else if(arg=="Window Data:Yes"){
    //Windowing Data
    windowdata=true;
    help.appendText(arg+"\n");
    help.appendText("Choose Amplitude Scale\n");
} else if(arg=="Window Data:No"){
    windowdata=false;
help.appendText(arg+"\n");
help.appendText("Choose Amplitude Scale\n");
} else if(tmp1=="Amplitude Scale"){
    //Amplitude Scale
    if(arg=="Reduce 10 times"){
        AmpScale=0.1;
        help.appendText(arg+" +tmp1+\n");
        help.appendText("Choose Water Elevation Scale\n");
    } else if(arg=="Reduce 7 times"){
        AmpScale=1.0/7.0;
        help.appendText(arg+" +tmp1+\n");
        help.appendText("Choose Water Elevation Scale\n");
    } else if(arg=="Reduce 5 times"){
        AmpScale=.2;
        help.appendText(arg+" +tmp1+\n");
        help.appendText("Choose Water Elevation Scale\n");
    } else if(arg=="Reduce 3 times"){
        AmpScale=1.0/3.0;
        help.appendText(arg+" +tmp1+\n");
        help.appendText("Choose Water Elevation Scale\n");
    } else if(arg=="Reduce 2 times"){
        AmpScale=0.5;
        help.appendText(arg+" +tmp1+\n");
        help.appendText("Choose Water Elevation Scale\n");
    } else if(arg=="No change"){
        AmpScale=1.0;
        help.appendText(arg+" +tmp1+\n");
        help.appendText("Choose Water Elevation Scale\n");
    } else if(arg=="Increase 2 times"){
        AmpScale=2.0;
        help.appendText(arg+" +tmp1+\n");
        help.appendText("Choose Water Elevation Scale\n");
    } else if(arg=="Increase 3 times"){
        AmpScale=3.0;
        help.appendText(arg+" +tmp1+\n");
        help.appendText("Choose Water Elevation Scale\n");
    } else if(arg=="Increase 5 times"){
        AmpScale=5.0;
        help.appendText(arg+" +tmp1+\n");
    } else if(tmp1=="Water Elevation Scale"){
        //Water Elevation Scale
        if(arg=="Reduce 10 times"){
            help.appendText(arg+" +tmp1+\n");
            help.appendText("Choose Water Elevation Scale\n");
        } else if(arg=="Reduce 7 times"){
            AmpScale=1.0/7.0;
            help.appendText(arg+" +tmp1+\n");
            help.appendText("Choose Water Elevation Scale\n");
        } else if(arg=="Reduce 5 times"){
            AmpScale=.2;
            help.appendText(arg+" +tmp1+\n");
            help.appendText("Choose Water Elevation Scale\n");
        } else if(arg=="Reduce 3 times"){
            AmpScale=1.0/3.0;
            help.appendText(arg+" +tmp1+\n");
            help.appendText("Choose Water Elevation Scale\n");
        } else if(arg=="Reduce 2 times"){
            AmpScale=0.5;
            help.appendText(arg+" +tmp1+\n");
            help.appendText("Choose Water Elevation Scale\n");
        } else if(arg=="No change"){
            AmpScale=1.0;
            help.appendText(arg+" +tmp1+\n");
            help.appendText("Choose Water Elevation Scale\n");
        } else if(arg=="Increase 2 times"){
            AmpScale=2.0;
            help.appendText(arg+" +tmp1+\n");
            help.appendText("Choose Water Elevation Scale\n");
        } else if(arg=="Increase 3 times"){
            AmpScale=3.0;
            help.appendText(arg+" +tmp1+\n");
            help.appendText("Choose Water Elevation Scale\n");
        } else if(arg=="Increase 5 times"){
            AmpScale=5.0;
            help.appendText(arg+" +tmp1+\n");
        }
    }
help.appendText("Choose Water Elevation Scale\n");
} else if(arg=="Increase 7 times"){
AmpScale=7.0;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Water Elevation Scale\n");
} else if(arg=="Increase 10 times"){
AmpScale=10.0;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Water Elevation Scale\n");
}
} else if(tmp1=="Water Elevation Scale"){
//Water Elev. Scale
if(arg=="Reduce 10 times"){
WaterScale=.1;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Frame Scale\n");
} else if(arg=="Reduce 7 times"){
WaterScale=1.0/7.0;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Frame Scale\n");
} else if(arg=="Reduce 5 times"){
WaterScale=0.2;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Frame Scale\n");
} else if(arg=="Reduce 3 times"){
WaterScale=1.0/3.0;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Frame Scale\n");
} else if(arg=="Reduce 2 times"){
WaterScale=0.5;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Frame Scale\n");
} else if(arg=="No change"){
WaterScale=1.0;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Frame Scale\n");
} else if(arg=="Increase 2 times"){
WaterScale=2.0;
help.appendText(arg+" "+tmp1+"\n");
help.appendText("Choose Frame Scale\n");
} else if(arg=="Increase 3 times"){
    WaterScale=3.0;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Frame Scale\n");
} else if(arg=="Increase 5 times"){
    WaterScale=5.0;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Frame Scale\n");
} else if(arg=="Increase 7 times"){
    WaterScale=7.0;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Frame Scale\n");
} else if(arg=="Increase 10 times"){
    WaterScale=10.0;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Frame Scale\n");
}
} else if(arg=="Frame Reduce 10 times"){
    FrameScale=0.1;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Adjust Gray Value\n");
} else if(arg=="Frame Reduce 8 times"){
    FrameScale=1.0/8.0;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Adjust Gray Value\n");
} else if(arg=="Frame Reduce 5 times"){
    FrameScale=0.2;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Adjust Gray Value\n");
} else if(arg=="Frame Reduce 3 times"){
    FrameScale=1.0/3.0;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Adjust Gray Value\n");
} else if(arg=="Frame Reduce 2 times"){
    FrameScale=0.5;
    help.appendText(arg+" "+tmp1+"\n");
    help.appendText("Choose Adjust Gray Value\n");
}