FFTLab (1C)

Young Won Lim
5/12/14

this document is based on
http://sepwww.stanford.edu/oldsep/hale/FftLab.html

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Data Processing & Graphic Parts

- MainPanel
- ComplexSamplesPanel
- SamplesPanel
- SamplesView [1-3]
- DrawOneSample()
- UpdateDrawingSize()
- SetSampleValue()
- FftLabController [1-4]
- Samples
- Fft
- ControlPanel [1-2]

MainPanel, ComplexSamplesPanel

```
class MainPanel extends Panel {

   public MainPanel(SamplesView fRealView,
                    SamplesView fImagView,
                    SamplesView gRealView,
                    SamplesView gImagView) {
      setLayout(new GridLayout(2,1,1,1));
      add(new ComplexSamplesPanel(fRealView,
                    fImagView,
                    "f(x)"));
      add(new ComplexSamplesPanel(gRealView,
                    gImagView,
                    "F(k)"));
   }

   public void paint(Graphics g) {
      Dimension d = size();
      g.setColor(Color.blue);
      g.draw3DRect(0,0,d.width-1,d.height-1,true);
   }

   public Insets insets() {
      return new Insets(1,1,1,1);
   }
}
```

```
class ComplexSamplesPanel extends Panel {

   public ComplexSamplesPanel(SamplesView realView,
                              SamplesView imagView,
                              String label) {
      setLayout(new BorderLayout());
      add("North",new Label(label,Label.CENTER));
      Panel panel = new Panel();
      panel.setLayout(new GridLayout(1,2,1,1));
      panel.add(new SamplesPanel(realView,"Real"));
      panel.add(new SamplesPanel(imagView,"Imaginary"));
      add("Center",panel);
   }

   public void paint(Graphics g) {
      Dimension d = size();
      g.setColor(Color.blue);
      g.draw3DRect(0,0,d.width-1,d.height-1,true);
   }

   public Insets insets() {
      return new Insets(1,1,1,1);
   }
}
```

SamplesPanel

```
class SamplesPanel extends Panel {

    public SamplesPanel(SamplesView view, String label) {
        setLayout(new BorderLayout());
        add("North",new Label(label,Label.CENTER));
        add("Center",view);
    }

    public void paint(Graphics g) {
        Dimension d = size();
        g.setColor(Color.blue);
        g.draw3DRect(0,0,d.width-1,d.height-1,true);
    }

    public Insets insets() {
        return new Insets(1,1,1,1);
    }
}
```

SamplesView (1)

```
class SamplesView extends Canvas {

  public static int EDIT_NONE     = 0;
  public static int EDIT_DRAW     = 1;
  public static int EDIT_ZERO     = 2;
  public static int EDIT_NEGATE   = 3;
  public static int EDIT_SHIFT    = 4;

  public Samples samples;

  public SamplesView(Samples s) {
     samples = s;
     setSampleValue(1.0f);
     updateDrawingSizes();
     setBackground(Color.yellow);
  }

  public void setSampleValue(float v) {
     int height = size().height;
     sampleValue = (v!=0.0f)?v:1.0f;
     sampleScale = -0.25f*height/sampleValue;
  }

  public void setEditMode(int mode) {
     editMode = mode;
  }

  public void paint(Graphics g) {
     updateDrawingSizes();
     drawSamples(g);
  }
```

```
// class SamplesView extends Canvas { // 2 of 6

  public Dimension minimumSize() {
     return new Dimension(100,50);
  }

  public Dimension preferredSize() {
     return minimumSize();
  }

  public boolean mouseDown(Event e, int x, int y) {
     if (editMode==EDIT_NONE) return true;
     lastDrag = -1;
     return mouseDrag(e,x,y);
  }
```

SamplesView (2)

// class **SamplesView** extends Canvas { // 3 of 6

```
public boolean mouseDrag(Event e, int x, int y) {
   if (editMode==EDIT_NONE) return true;

   if (x<0 || x>size().width) return true;
   if (y<sampleRadius || y>size().height-sampleRadius)
     return true;

   int i = (int)((float)(x-sampleStart)/(float)sampleWidth+0.5);
   if (i<0) i = 0;
   if (i>=samples.values.length) i = samples.values.length-1;
   if (editMode==EDIT_NEGATE && i==lastDrag) return true;

   Graphics g = getGraphics();


   if (editMode==EDIT_SHIFT) {
      if (i != lastDrag && lastDrag>=0) {
         g.setColor(getBackground());
         drawSamples(g);

         samples.rotate(i-lastDrag);

         g.setColor(getForeground());
         drawSamples(g);
      }
      lastDrag = i;
      return true;
   }
```

// class **SamplesView** extends Canvas { // 4 of 6

```
      g.setColor(getBackground());
      drawOneSample(g,i);

      if (editMode==EDIT_ZERO) {
         samples.values[i] = 0.0f;
      } else if (editMode==EDIT_NEGATE) {
         samples.values[i] = -samples.values[i];
      } else {
         samples.values[i] = (float)(y-sampleBase) / sampleScale;
      }

      g.setColor(getForeground());
      drawOneSample(g,i);

      lastDrag = i;

      return true;
   }
```

SamplesView (3)

```
// class SamplesView extends Canvas { // 5 of 6

public boolean mouseUp(Event e, int x, int y) {
      if (editMode!=EDIT_NONE) samples.notifyObservers();
      return true;
   }

   private int editMode = EDIT_DRAW;
   private int  sampleStart, sampleBase,
                ampleWidth, sampleRadius;
   private float sampleScale,sampleValue;
   private int lastDrag;

   private void drawOneSample(Graphics g, int i) {
      int x = sampleStart+i*sampleWidth;
      int y = sampleBase;
      int r = sampleRadius;
      int w = sampleWidth;
      int h = (int)(samples.values[i]*sampleScale);
      g.drawLine(x-w/2,y, x+w/2,y);
      g.drawLine(x,y, x,y+h);
      if (i==samples.origin) {
         g.drawOval(x-r, y+h-r, 2*r, 2*r);
      } else {
         g.fillOval(x-r, y+h-r, 2*r+1, 2*r+1);
      }
   }
```

```
// class SamplesView extends Canvas { // 6 of 6

   private void drawSamples(Graphics g) {
      for (int i=0; i<samples.values.length; ++i) {
         drawOneSample(g,i);
      }
   }

   private void updateDrawingSizes() {
      int width = size().width;
      int height = size().height;
      int nSamples = samples.values.length;
      sampleWidth = (int)((float)width/(float)(nSamples+1));
      sampleStart = (width-(nSamples-1)*sampleWidth)/2;
      sampleBase = (int)(0.5f*height);
      sampleScale = -0.25f*height/sampleValue;
      sampleRadius = (int)(0.4f*sampleWidth);
      int maxRadius = (int)(0.5f*height);
      if (sampleRadius>maxRadius)
        sampleRadius = maxRadius;
   }
}
```

DrawOneSample()

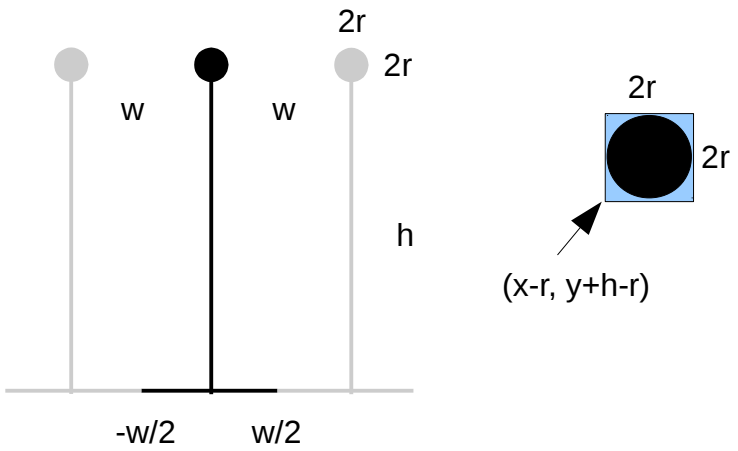
void setColor(Color color)
void setFont(Font font)
void drawString(String text, int x, int y)
void drawLine(int x1, int y1, int x2, int y2)
void drawRect(int x, int y, int width, int height)
void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
void drawOval(int x, int y, int width, int height)
void drawArc(int x, int y, int width, int height, int startAngle, int angularExtent)
void fillRect(int x, int y, int width, int height)
void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
void fillOval(int x, int y, int width, int height)
void fillArc(int x, int y, int width, int height, int startAngle, int angularExtent)

mouseDrag

drawSamples

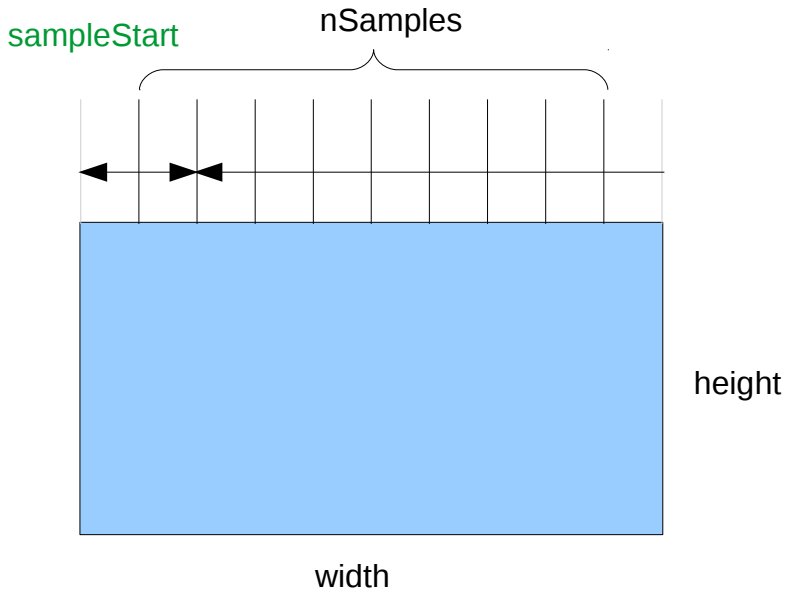
drawOneSample

drawOneSample




```
private void drawOneSample(Graphics g, int i) {
    int x = sampleStart+i*sampleWidth;
    int y = sampleBase;
    int r = sampleRadius;
    int w = sampleWidth;
    int h = (int)(samples.values[i]*sampleScale);
    g.drawLine(x-w/2,y, x+w/2,y);
    g.drawLine(x,y, x,y+h);
    if (i==samples.origin) {
        g.drawOval(x-r, y+h-r, 2*r, 2*r);
    } else {
        g.fillOval(x-r, y+h-r, 2*r+1, 2*r+1);
    }
}
```

updateDrawingSizes()




```
private void updateDrawingSizes() {
    int width = size().width;
    int height = size().height;
    int nSamples = samples.values.length;
    sampleWidth = (int)((float)width/(float)(nSamples+1));
    sampleStart = (width-(nSamples-1)*sampleWidth)/2;
    sampleBase = (int)(0.5f*height);
    sampleScale = -0.25f*height/sampleValue;
    sampleRadius = (int)(0.4f*sampleWidth);
    int maxRadius = (int)(0.5f*height);
    if (sampleRadius>maxRadius)
      sampleRadius = maxRadius;
}
```

setSampleValue()

```
class FftLabController implements Observer {

    private void updateSampleValues(SamplesView realView, SamplesView imagView) {
        float sv = computeSampleValue(realView.samples,imagView.samples);
        realView.setSampleValue(sv);
        imagView.setSampleValue(sv);
    }

    private float computeSampleValue(Samples real, Samples imag) {
        float sv = 0.0f;
        float v[];
        v = real.values;
        for (int i=0; i<length; ++i) {
            float si = v[i];
            if (-si>sv) sv = -si;
            else if (si>sv) sv = si;
        }
        v = imag.values;
        for (int i=0; i<length; ++i) {
            float si = v[i];
            if (-si>sv) sv = -si;
            else if (si>sv) sv = si;
        }
        return sv;
    }


}
```

sv = max{ | v[i] | }

```
class SamplesView extends Canvas {

    public SamplesView(Samples s) {
        samples = s;
        setSampleValue(1.0f);
        updateDrawingSizes();
        setBackground(Color.yellow);
    }

    public void setSampleValue(float v) {
        int height = size().height;
        sampleValue = (v!=0.0f) ? v : 1.0f;
        sampleScale = -0.25f*height/sampleValue;
    }
```

max{ | v[i] | }

max{ | v[i] | }
corresponds to -h/4

FftLabController (1)

```
class FftLabController implements Observer {

   public Samples      fReal, fImag, gReal, gImag;
   public SamplesView  fRealView, fImagView,
                       gRealView, gImagView;

   public FftLabController() {

      int origin = (originCentered)? length/2 : 0;
      fReal = new Samples(length,origin);
      fImag = new Samples(length,origin);
      gReal = new Samples(length,origin);
      gImag = new Samples(length,origin);
      initSamples();

      fReal.addObserver(this);
      fImag.addObserver(this);
      gReal.addObserver(this);
      gImag.addObserver(this);

      fRealView = new SamplesView(fReal);
      fImagView = new SamplesView(fImag);
      gRealView = new SamplesView(gReal);
      gImagView = new SamplesView(gImag);
      updateSampleValues(fRealView,fImagView);
      updateSampleValues(gRealView,gImagView);
   }

   public int getEditMode() {
      return editMode;
   }
```

```
// class FftLabController implements Observer { // 2 of 7

   public int getLength() {
      return length;
   }

   public void setLength(int length) {
      this.length = length;
      updateLengths();
      updateOrigins();
      initSamples();
      updateSampleValues(fRealView,fImagView);
      updateSampleValues(gRealView,gImagView);
      repaintViews();
   }

   public boolean getOriginCentered() {
      return originCentered;
   }

   public void setOriginCentered(boolean centered) {
      if (centered==originCentered) return;
      originCentered = centered;
      updateOrigins();
      repaintViews();
   }
```

FftLabController (2)

// class **FftLabController** implements Observer { // 3 of 7

```
public void zeroAll() {
   fReal.zero();
   fImag.zero();
   gReal.zero();
   gImag.zero();
   repaintViews();
}

public void update(Observable o, Object arg) {
   Samples s = (Samples)o;
   if (s==fReal || s==fImag) {
      transform(1,fReal,fImag,gReal,gImag);
      updateSampleValues(gRealView,gImagView);
      gRealView.repaint();
      gImagView.repaint();
   } else {
      transform(-1,gReal,gImag,fReal,fImag);
      updateSampleValues(fRealView,fImagView);
      fRealView.repaint();
      fImagView.repaint();
   }
}
```

```
private int editMode = SamplesView.EDIT_DRAW;
private int length = 32;
private boolean originCentered = false;
```

// class **FftLabController** implements Observer { // 4 of 7

```
private float computeSampleValue
                    (Samples real, Samples imag) {
   float sv = 0.0f;
   float v[];
   v = real.values;
   for (int i=0; i<length; ++i) {
      float si = v[i];
      if (-si>sv) sv = -si;
      else if (si>sv) sv = si;
   }
   v = imag.values;
   for (int i=0; i<length; ++i) {
      float si = v[i];
      if (-si>sv) sv = -si;
      else if (si>sv) sv = si;
   }
   return sv;
}

private void updateSampleValues
     (SamplesView realView, SamplesView imagView) {
   float sv = computeSampleValue
            (realView.samples,imagView.samples);
   realView.setSampleValue(sv);
   imagView.setSampleValue(sv);
}
```

FftLabController (3)

// class **FftLabController** implements Observer { // 5 of 7

```
private void transform(int sign,
                       Samples sar, Samples sai,
                       Samples bar, Samples bai) {
   float ar[] = sar.values;
   float ai[] = sai.values;
   float br[] = bar.values;
   float bi[] = bai.values;

   for (int i=0; i<length; ++i) {
      br[i] = ar[i];
      bi[i] = ai[i];
   }

   if (originCentered) {
      for (int i=1; i<length; i+=2) {
            br[i] = -br[i];
            bi[i] = -bi[i];
      }
   }

   Fft.complexToComplex(sign,length,br,bi);

   if (originCentered) {
      for (int i=1; i<length; i+=2) {
            br[i] = -br[i];
            bi[i] = -bi[i];
      }
   }
}
```

// class **FftLabController** implements Observer { // 6 of 7

```
private void initSamples() {
   fReal.values[fReal.origin+1] = 1.0f;
   transform(1,fReal,fImag,gReal,gImag);
}

private void updateLengths() {
   int length = this.length;
   fReal.setLength(length);
   fImag.setLength(length);
   gReal.setLength(length);
   gImag.setLength(length);
}

private void updateOrigins() {
   int origin = (originCentered)?length/2:0;
   int shift = origin-fReal.origin;
   fReal.origin = origin;
   fImag.origin = origin;
   gReal.origin = origin;
   gImag.origin = origin;
   fReal.rotate(shift);
   fImag.rotate(shift);
   gReal.rotate(shift);
   gImag.rotate(shift);
}
```

FftLabController (4)

```
// class FftLabController implements Observer { // 7 of 7

    private void repaintViews() {
        fRealView.repaint();
        fImagView.repaint();
        gRealView.repaint();
        gImagView.repaint();
    }

    private void shiftSamples(Samples s, int shift) {
        float temp[] = new float[length];
        int j = shift%length;
        for (int i=0; i<length; ++i,++j) {
            if (j<0) j += length;
            if (j>=length) j -= length;
            temp[j] = s.values[i];
        }
        s.values = temp;
    }

}
```

Samples

```
class Samples extends Observable {

    public float values[];
    public int origin;

    public Samples(int length, int origin) {
        this.origin = origin;
        values = new float[length];
        zero();
    }

    public void setLength(int length) {
        if (length==values.length) return;
        values = new float[length];
        zero();
    }

    public void zero() {
        for (int i=0; i<values.length; ++i) values[i] = 0.0f;
    }

    public void rotate(int n) {
        int length = values.length;
        float temp[] = new float[length];

        int j = n % length;
        for (int i=0; i<length; ++i,++j) {
            if (j<0) j += length;
            if (j>=length) j -= length;
            temp[j] = values[i];
        }
        values = temp;
    }

    public void notifyObservers() {
        setChanged();
        super.notifyObservers();
    }
}
```

Fft

```
class Fft {
   public static void complexToComplex
                (int sign, int n, float ar[], float ai[]) {
      float scale = (float)Math.sqrt(1.0f/n);

      int i,j;
      for (i=j=0; i<n; ++i) {
         if (j>=i) {
                float tempr = ar[j]*scale;
                float tempi = ai[j]*scale;
                ar[j] = ar[i]*scale;
                ai[j] = ai[i]*scale;
                ar[i] = tempr;
                ai[i] = tempi;
         }
         int m = n/2;
         while (m>=1 && j>=m) {
                j -= m;
                m /= 2;
         }
         j += m;
      }
      int mmax,istep;
      for (mmax=1,istep=2*mmax; mmax<n; mmax=istep,istep=2*mmax) {
         float delta = (float)sign*3.141592654f/(float)mmax;
         for (int m=0; m<mmax; ++m) {
                float w = (float)m*delta;
                float wr = (float)Math.cos(w);
                float wi = (float)Math.sin(w);
                for (i=m; i<n; i+=istep) {
                   j = i+mmax;
                   float tr = wr*ar[j]-wi*ai[j];
                   float ti = wr*ai[j]+wi*ar[j];
                   ar[j] = ar[i]-tr;
                   ai[j] = ai[i]-ti;
                   ar[i] += tr;
                   ai[i] += ti;
                }
         }
         mmax = istep;
      }
   }
}
```

ControlPanel (1)

```
class ControlPanel extends Panel {

   public ControlPanel(FftLabController c) {
      this.c = c;
      add(new Checkbox("Origin Centered"));

      length = new LabeledChoice("Length:");
      length.choice.addItem("16");
      length.choice.addItem("32");
      length.choice.addItem("64");
      length.choice.select("32");
      add(length);

      mode = new LabeledChoice("Editing:");
      mode.choice.addItem("Draw");
      mode.choice.addItem("Negate");
      mode.choice.addItem("Zero");
      mode.choice.addItem("Shift");
      mode.choice.addItem("None");
      mode.choice.select("Draw");
      add(mode);

      add(new Button("Zero All"));
   }
```

```
public void paint(Graphics g) {
   Dimension d = size();
   g.setColor(Color.blue);
   g.draw3DRect(0,0,d.width-1,d.height-1,true);
}

public Insets insets() {
   return new Insets(1,1,1,1);
}
```

ControlPanel (2)

```
    public boolean handleEvent(Event e) {

        if (e.target instanceof Button) {
            c.zeroAll();
            return true;
        } else if (e.target instanceof Checkbox) {
            Checkbox cb = (Checkbox)e.target;
            c.setOriginCentered(cb.getState());
            return true;
        } else if (e.target instanceof Choice) {
                                                    <-
        }
        return false;
    }

    private FftLabController c;
    private LabeledChoice length;
    private LabeledChoice mode;
}
```

```
        if (e.target==length.choice) {
            String item = length.choice.getSelectedItem();

            c.setLength(Integer.parseInt(item));
            return true;

        } else if (e.target==mode.choice) {
            String item = mode.choice.getSelectedItem();

            if (item=="None") {
                c.setEditMode(SamplesView.EDIT_NONE);
            } else if (item=="Draw") {
                c.setEditMode(SamplesView.EDIT_DRAW);
            } else if (item=="Negate") {
                c.setEditMode(SamplesView.EDIT_NEGATE);
            } else if (item=="Zero") {
                c.setEditMode(SamplesView.EDIT_ZERO);
            } else if (item=="Shift") {
                c.setEditMode(SamplesView.EDIT_SHIFT);
            }
            return true;
        }
```

Class Structure

func1A()

```
setLayout(new BorderLayout());
add("Center",mainPanel);
add("South",controlPanel);
Frame frame = new
Frame("FFT Laboratory");
frame.add("Center",fftLab);
frame.resize(600,400);
frame.show();
add(new ComplexSamplesPanel(fRealView,fImagView,"f(x)"));
add(new ComplexSamplesPanel(gRealView,gImagView,"F(k)"));
setLayout(new BorderLayout());
add("North",new Label(label,Label.CENTER));
add("Center",view);
```

Swing Graphics Class

func1A()

```
g.setColor(Color.blue);
g.draw3DRect(0,0,d.width-1,d.height-1,true);
g.setColor(Color.blue);
g.draw3DRect(0,0,d.width-1,d.height-1,true);
g.setColor(Color.blue);
g.draw3DRect(0,0,d.width-1,d.height-1,true);
      g.setColor(getBackground());
      g.setColor(getForeground());
g.setColor(getBackground());
g.setColor(getForeground());
g.drawLine(x-w/2,y,x+w/2,y);
g.drawLine(x,y,x,y+h);
   g.drawOval(x-r,y+h-r,2*r,2*r);
   g.fillOval(x-r,y+h-r,2*r+1,2*r+1);
```

FftLabController (4)

Samples

```
class Samples extends Observable {
                                        public void notifyObservers() {
                                            setChanged();
                                            super.notifyObservers();
                                        }
                                    }
```

References

[1] Java in a nutshell, 4th ed, David Flanagan
[2] An Introduction to Object-Oriented Programming with Java, C. Thomas, Wu
[3] Power Java, I. K. Chun (in Korean)
[4] http://sepwww.stanford.edu/oldsep/hale/FftLab.html

SamplesView (2)

```
class MainPanel extends Panel {
   public MainPanel(      SamplesView fRealView,
                          SamplesView fImagView,
                          SamplesView gRealView,
                          SamplesView gImagView) {
   … }
}

class ComplexSamplesPanel extends Panel {
   public ComplexSamplesPanel(
               SamplesView realView,
               SamplesView imagView,
               String label) {
   … }
}

class SamplesPanel extends Panel {
   public SamplesPanel(SamplesView view, String label) {
   ...   }
}
```

```
class ControlPanel extends Panel {
  …
  public boolean handleEvent(Event e) {
  …
            if (item=="None") {
               c.setEditMode(SamplesView.EDIT_NONE);
            } else if (item=="Draw") {
               c.setEditMode(SamplesView.EDIT_DRAW);
            } else if (item=="Negate") {
              c.setEditMode(SamplesView.EDIT_NEGATE);
            } else if (item=="Zero") {
               c.setEditMode(SamplesView.EDIT_ZERO);
            } else if (item=="Shift") {
               c.setEditMode(SamplesView.EDIT_SHIFT);
            }
    … }
}
```

SamplesView (3)

```
class FftLabController implements Observer {
   …
   public SamplesView     fRealView, fImagView,
                          gRealView, gImagView;

   public FftLabController() {
      ,,,
      fRealView = new SamplesView(fReal);
      fImagView = new SamplesView(fImag);
      gRealView = new SamplesView(gReal);
      gImagView = new SamplesView(gImag);
      …
   }

   private void updateSampleValues
     (SamplesView realView, SamplesView imagView) {
      float sv = computeSampleValue
                 (realView.samples,imagView.samples);
      realView.setSampleValue(sv);
      imagView.setSampleValue(sv);
   }
```

```
class ControlPanel extends Panel {
  …
  public boolean handleEvent(Event e) {
  …
              if (item=="None") {
                 c.setEditMode(SamplesView.EDIT_NONE);
              } else if (item=="Draw") {
                 c.setEditMode(SamplesView.EDIT_DRAW);
              } else if (item=="Negate") {
                 c.setEditMode(SamplesView.EDIT_NEGATE);
              } else if (item=="Zero") {
                 c.setEditMode(SamplesView.EDIT_ZERO);
              } else if (item=="Shift") {
                 c.setEditMode(SamplesView.EDIT_SHIFT);
              }
    … }
}
```