```
   double[][] realPhasor, imagPhasor, amplitude;
   int n;    // grid points on a side
   double a; // side length

   public HuygensApp() {
      // interpolated plot looks best
      frame.convertToInterpolatedPlot();
      frame.setPaletteType(ColorMapper.RED);
      frame.setInteractiveMouseHandler(this);
   }

   public void initialize() {
      n = control.getInt("grid size");
      a = control.getDouble("length");
      frame.setPreferredMinMax(-a/2, a/2, -a/2, a/2);
      realPhasor = new double[n][n];
      imagPhasor = new double[n][n];
      amplitude = new double[n][n];
      frame.setAll(amplitude);
      initPhasors();
   }

   void initPhasors() {
      for(int ix = 0;ix<n;ix++) {
         for(int iy = 0;iy<n;iy++) {
            // zero the phasor
            imagPhasor[ix][iy] = realPhasor[ix][iy] = 0;
         }
      }
      // an iterator for the sources in the frame
      Iterator it = frame.getDrawables().iterator(); // source iterator
      // counts the number of sources
      int counter = 0;
      while(it.hasNext()) {
         InteractiveShape source = (InteractiveShape) it.next();
         counter++;
         double xs = source.getX(), ys = source.getY();
         for(int ix = 0;ix<n;ix++) {
            double x = frame.indexToX(ix);
            // source->gridpoint
            double dx = (xs-x);
            for(int iy = 0;iy<n;iy++) {
               double y = frame.indexToY(iy);
               // charge->gridpoint
               double dy = (ys-y);
               double r = Math.sqrt(dx*dx+dy*dy);
               realPhasor[ix][iy] += (r==0) ? 0 : Math.cos(PI2*r)/r;
               imagPhasor[ix][iy] += (r==0) ? 0 : Math.sin(PI2*r)/r;
            }
         }
      }
      double cos = Math.cos(-PI2*time);
      double sin = Math.sin(-PI2*time);
      for(int ix = 0;ix<n;ix++) {
         for(int iy = 0;iy<n;iy++) {
            // only the real part of the complex field is physical
```

```
            double re = cos*realPhasor[ix][iy]-sin*imagPhasor[ix][iy];
            amplitude[ix][iy] = re*re;
         }
      }
      frame.setZRange(false, 0, 0.2*counter); // scale the intensity
      frame.setAll(amplitude);
   }

   public void reset() {
      time = 0;
      control.setValue("grid size", 128);
      control.setValue("length", 10);
      frame.clearDrawables();
      frame.setMessage("t = "+decimalFormat.format(time));
      control.println("Source button creates a new source.");
      control.println("Drag sources after they are created.");
      initialize();
   }

   public void createSource() {
      InteractiveShape ishape =
           InteractiveShape.createCircle(0, 0, 0.5);
      frame.addDrawable(ishape);
      initPhasors();
      frame.repaint();
   }

   public void handleMouseAction(InteractivePanel panel,
                                 MouseEvent evt) {
      panel.handleMouseAction(panel, evt); // panel moves the source
      if(panel.getMouseAction()==InteractivePanel.MOUSE_DRAGGED) {
         initPhasors();
      }
   }

   protected void doStep() {
      time += 0.1;
      double cos = Math.cos(-PI2*time);
      double sin = Math.sin(-PI2*time);
      for(int ix = 0;ix<n;ix++) {
         for(int iy = 0;iy<n;iy++) {
            double re = cos*realPhasor[ix][iy]-sin*imagPhasor[ix][iy];
            amplitude[ix][iy] = re*re;
         }
      }
      frame.setAll(amplitude);
      frame.setMessage("t="+decimalFormat.format(time));
   }

   public static void main(String[] args) {
      OSPControl control =
           SimulationControl.createApp(new HuygensApp());
      control.addButton("createSource", "Source");
   }
}
```