Programme de la Web-Cam

Cam.java

```
import java.awt.*;
import java.lang.*;

public class cam
{

    public static void main(String[] args)
    {
      mainWindow mWindow = new mainWindow();
    }

}
```

CameraImage.java

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\
 * This file is part of CMUcamGUI, a java program that helps             *
 * interface with the CMUcam Vision Board.                               *
 * Contact cmucam@cs.cmu.edu, or see                                     *
 * http://www.cs.cmu.edu/~cmucam for more information.                   *
 *                                                                       *
 * Copyright 2001 Anthony Rowe                                           *
 *                                                                       *
 * This program is free software; you can redistribute it and/or         *
 * modify it under the terms of the GNU General Public License           *
 * as published by the Free Software Foundation - version 2.             *
 *                                                                       *
 * This program is distributed in the hope that it will be               *
 * useful, but WITHOUT ANY WARRANTY; without even the implied            *
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR               *
 * PURPOSE.  See the GNU General Public License in file COPYING          *
 * for more details.                                                     *
 *                                                                       *
 * You should have received a copy of the GNU General Public             *
 * License along with this program; if not, write to the Free            *
 * Software Foundation, Inc., 59 Temple Place - Suite 330,               *
 * Boston, MA 02111, USA.                                                *
\* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;

/*
This is where most of the good stuff happens.  I'm personally not a
big fan of oop, so don't blame me for the mess...(java is just nice for
rapid GUI prototyping)

This object gets called from mainWindow when menu events happen.
This then holds the current on screen image and does the actual
double buffered drawing to the window.
```

```
It handles the following:
1) Track Color box
2) Middle Mass point
3) Bitmap for track color
4) lines for lm mean
5) Dump Frame
*/
public class CameraImage extends Canvas implements Serializable
{
    Frame picker_f;
    boolean pickerStat;

    public int mm;
    int mmx,mmy,mmoldx,mmoldy;
    char bmask[][];
    int lm,lmIndex,tc;
    int width,height;
    int pixels[];
    int col,row;
    int
mx,my,xcoord,ycoord,zcoord,oldz,oldx,oldy,x2coord,y2coord,oldx2,oldy2,conf;
    MemoryImageSource source;
    Image image;
    Image offScreenImage;
    Graphics offScreenGC;
    zoomBox azoomBox;
    Label red_l,green_l,blue_l;
    Panel colorBox;
    int frames,fps;
    long startTime;
    int start;
    int scale;
    long wdTimer;
    colorTrack myTrack;
    outWindow outWin;
    meanWindow myMean;
    rawWindow myRaw;
    channelWindow myChannels;
    setWindow mySW;
    camSettings myCamSettings;
    Frame channel_f;
    serialComm mySerial;
    CameraImage(int x,int y,String commPort)
    {
      // This sets up the serial communication object
      // all of the read/write commands get called from here
      mySerial=new serialComm(commPort);
      // Lets clear some variables...
      bmask= new char[10][48];
      lmIndex=0;
      mmoldx=0;
      mmoldy=0;
      mmx=0;
      mmy=0;
      mm=1;
      lm=0;
      scale=0;
      start=1;
      zcoord=0;
      xcoord=0;
      ycoord=0;
```

```
x2coord=0;
y2coord=0;
oldx=0;
oldy=0;
oldx2=0;
oldy2=0;
oldz=0;
conf=0;
tc=0;
pickerStat=false;
picker_f= new Frame("Color Picker");
Panel picker_p = new Panel();
red_l = new Label("0");
green_l = new Label("0");
blue_l = new Label("0");
azoomBox = new zoomBox();
colorBox = new Panel();
colorBox.setSize(20,20);
picker_f.setBackground(Color.lightGray);
picker_p.setLayout(new GridLayout(2,3));
picker_p.add(new Label("Red"));
picker_p.add(new Label("Green"));
picker_p.add(new Label("Blue"));
picker_p.add(red_l);
picker_p.add(green_l);
picker_p.add(blue_l);
picker_f.add("North",new Panel());
picker_f.add("South",new Panel());
picker_f.add("West",new Panel());
picker_f.add("Center",colorBox);
picker_f.add("East",picker_p);
picker_f.setSize(200,70);
picker_f.setLocation(260,0);
picker_f.show();
pickerStat=true;
setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR_CURSOR));
enableEvents(AWTEvent.MOUSE_EVENT_MASK |
           AWTEvent.MOUSE_MOTION_EVENT_MASK);
col=0;
row=0;
this.setBackground(Color.black);
width=x;
height=y;
pixels=new int[width*height];
source = new MemoryImageSource(width,height,pixels,0,width);
source.setAnimated(true);
image=createImage(source);
frames=0;
      fps=0;
Date time = new Date();
      startTime=time.getTime();

myTrack = new colorTrack();
outWin = new outWindow();
myMean = new meanWindow();
myChannels = new channelWindow(width,height);
mySW = new setWindow();
myCamSettings = new camSettings();
channel_f= new Frame("Image Channels");
channel_f.add(myChannels);
channel_f.setBackground(Color.lightGray);
```

```
  channel_f.setSize(200,500);
  channel_f.setLocation(670,0);
  channel_f.show();
}
public void hideSW() {mySW.hide();}
public void showSW(){mySW.show();}
public void hideTrack() {myTrack.hide();}
public void showTrack(){myTrack.show();}
public void hideMean() {myMean.hide();}
public void showMean(){myMean.show();}
public void processEvent(AWTEvent e)
{
  mx=((MouseEvent)e).getX();
  my=((MouseEvent)e).getY();
  int tx,ty,pix,tmp;
  tx=mx-(getSize().width-width)/2;
  ty=my-(getSize().height-height)/2;
  if(tx<0 || ty<0 || tx>width-1 || ty>height-1)
      pix=0;
  else
      pix=pixels[ty*width+tx];
  Color pixColor= new Color(pix);
  colorBox.setBackground(pixColor);
  tmp=pix & 0xFF;
  blue_l.setText(String.valueOf(tmp));
  tmp=pix>>8 & 0xFF;
  green_l.setText(String.valueOf(tmp));
  tmp=pix>>16 & 0xFF;
  red_l.setText(String.valueOf(tmp));
}



/*
  Set the window.
  Set the background image correctly for dump frame
*/
public int sw()
{
  for(int l=0; l<width; l++ )
      for(int d=0; d<height; d++ )
        pixels[d*width+l] = 0;
  sendCommand(mySW.sendString());
  source.newPixels(0,0,width,height);
return 0;
}
public int trackColor(int mode)
{
  int data;
  if(start==1)
      {

        if(mode==0)
            {
              if(!sendCommand(myTrack.sendString()))return 0;
            }
        else
            {
              if(!sendCommand("tw" + "\r")) return 0;
            }
      }
```

```
      try{
          data=mySerial.readByte();
          if(data==0xFE)  // Ignore a line mode mean packet
            {
                data=mySerial.readByte();
                while(data!=0xFD) data=mySerial.readByte();
            }
          if(data=='C' || data=='M' || data==0xAA)
            {
                tc=1;

                if(data==0xAA)
                  lm=1;
                else
                  lm=0;
                lmIndex=0;
                if(lm==1)  // Read in the bitmap for lm
                  {
                            try{
                              data=0;
                              int index=0;
                              Date time = new Date();
                              wdTimer=time.getTime();
                              while(data!=0xAA)
                                  {
                                    data=mySerial.readByte();
//(char)sPortIn.read();
                                    bmask[index][lmIndex]=(char)data;
                                    index++;
                                    if(index==10)
                                        {
                                          index=0;
                                          lmIndex++;
                                        }
                                          Date time2 = new Date();
                                          if(time2.getTime()-wdTimer>1000)
{System.out.println("<Track> time out"); return 0; }
                                  }


      data=mySerial.readByte();//(char)sPortIn.read();
                               if(data!=0xAA)
                                  {
                                    System.out.println( "Line Mode
Termination Error" );
                                  }
                              } catch(Exception e) {}
                              data=mySerial.readByte();
                              data=mySerial.readByte();
                              if(data=='S')
                                return 2;

                  }
                if(data=='M') // If middle mass is on
                  mm=1;
                else
                  mm=0;
                data=mySerial.readByte();
                if(mm==1)
                  {
                      mmx=mySerial.readNum();
```

```
                    mmy=mySerial.readNum();
                  }
                xcoord=mySerial.readNum();
                ycoord=mySerial.readNum();
                x2coord=mySerial.readNum();
                y2coord=mySerial.readNum();
                zcoord=mySerial.readNum();
                conf=mySerial.readNum();
                frames++;
                Date time = new Date();
                if( time.getTime()-startTime>1000 )
                  {
                    fps = frames;
                    frames=0;
                    startTime=time.getTime();
                  }

                /*if(lm==0)*/ outWin.append("x1="+ (new
Integer(xcoord)).toString()+" y1="+
                              (new Integer(ycoord)).toString()+" x2="+
                              (new Integer(x2coord)).toString()+" y2="+
                              (new Integer(y2coord)).toString()+" size="+
                              (new Integer(zcoord)).toString() + " fps="+
                               (new Integer(fps)).toString()+" conf="+
                              (new Integer(conf)).toString() );

                xcoord*=2;
                xcoord=160-xcoord;
                x2coord*=2;
                x2coord=160-x2coord;
                ycoord=144-ycoord;
                y2coord=144-y2coord;
                repaint();
            }
        }catch(Exception e) { System.out.println(e);}

      return 2;
    }


    // Set the camera
    public void setCamera()
    {

      sendCommand(myCamSettings.getString()+"\r");

    }
    /*
      this reads the current image and breaks
      it up into the channels window
    */
    public void splitChannels()
    {

      myChannels.splitChannels(pixels);

    }

public int getMean(int mode)
    {
      int data,cnt,pix;
```

```
      if(start==1)
          {
            if(!sendCommand("gm\r"))return 0;
          }

      cnt=0;
      data=mySerial.readByte();
      if(data==0xFE) // Start of line mode mean flag
          {
            int pcnt=0;
            tc=0;
            while(true)
                {
                  int red,green,blue;
                  red=mySerial.readByte();
                  if(red==0xFD) // Got the end of a packet
                      {
                        source.newPixels(0,0,width,height);
                        if(pcnt<143) scale=1;
                        else scale=0;
                        break;
                      }
                  green=mySerial.readByte();
                  if(green==0xFD) break;
                  blue=mySerial.readByte();
                  if(blue==0xFD) break;
                  pix=0;
                  pix+=blue;
                  pix+=green<<8;
                  pix+=red<<16;
                  Color pixColor= new Color(pix);
                  pcnt++;
                  // This loop draws the get mean lines on the background
image
                  for(int col=0; col<160; col++ )
                      {
                        if(cnt<143)
                            {
                              pixels[cnt*width+col] = pixColor.getRGB();
                              if(scale==1) pixels[(cnt+1)*width+col] =
pixColor.getRGB();
                            }

                      }

                  cnt++;
                  if(scale==1) cnt++;
                }

          }
      if(data=='S')  // Mean packet
            {
                int rm,gm,bm,rd,gd,bd;
                // Track Color Packet
                data=mySerial.readByte();
                rm=mySerial.readNum();
                gm=mySerial.readNum();
                bm=mySerial.readNum();
                rd=mySerial.readNum();
                gd=mySerial.readNum();
```

```
                bd=mySerial.readNum();

                myMean.update(rm,gm,bm,rd,gd,bd);
                frames++;
                Date time = new Date();
                if( time.getTime()-startTime>1000 )
                  {
                      fps = frames;
                      frames=0;
                      startTime=time.getTime();
                  }
                outWin.append( "C1m= " +rm+" C2m= " +bm+" C3m= "+gm+"
deviations= " + rd + "," + gd + "," + bd+ " fps= " + fps );
            }

      return 6;
    }




    /*
      Flushes all data out of the serial file buffer
    */
    public void flushBuf()
    {
      char a;
      try {
          while(mySerial.readNonBlock()!=0); // used to be an available
      } catch(Exception e) { System.out.println(e); }
    }

    /*
      Trys to get the camera to settle down and idle
    */
    public boolean idle()
    {
      char a,c,k,r,p;
      try
          {
              Date time = new Date();
            wdTimer=time.getTime();
            mySerial.writeStr("\r");
            while(true)
                {
                  a=mySerial.readByte();
                  if(a==':' )
                      {
                        start=1;
                        xcoord=0;
                        ycoord=0;
                        x2coord=0;
                        y2coord=0;
                        zcoord=0;

                        repaint();
                        outWin.append("Camera OK and idle..." );
                        return true;
                      }
                  Date time2 = new Date();
```

```
                if(time2.getTime()-wdTimer>1000)
{System.out.println("<idle> time out"); return false; }
                }


          }catch(Exception e) {System.out.println(e);}
      return false;

    }
    /*
      Writes the string to the output window
    */
    public void writeText(String str)
    {
      outWin.append( str );

    }
    /*
      Sends a command and checks for an ACK
      Returns 1 if ACK else 0
    */
    public boolean sendCommand(String command)
    {
      char a,c,k,r;
      a=0;
      try
          {
            //while(mySerial.readByte()!=0)
            //  {
            //      System.out.println( "Buffer" );
            //  }
            //while(mySerial.aval()!=0 )
            //   a=mySerial.readByte();
                //a=(char)sPortIn.read();
            mySerial.writeStr(command);
            a=mySerial.readByte();
            c=mySerial.readByte();
            k=mySerial.readByte();
            r=mySerial.readByte();
            if(a=='A' && c=='C' && k=='K' && r=='\r' )
                {
                  outWin.append( ">" + command.trim() + "< confirmed..." );
                  start=0;
                  return true;
                }
            else
                {
                  outWin.append( ">" + command.trim() +"< failed..." );
                  start=1;
                  //start=0;
                  //return true;
                  return false;
                }


          }catch(Exception e) {System.out.println(e);}
      return false;
    }

    /*
    Grabs the image column by column and displays it.
```

```
*/
public int dumpFrame()
{
  char a,c,k,r;
  int data, pix;


  if(start==1)
      if(!sendCommand("DF\r")) return 0;


  try{
      data=mySerial.readNonBlock();
      while(data!=0)
        {
        if(data==1)  // New frame
            {
              int tx,ty,tmp;
              outWin.append("Sending Frame,Please Wait..." );
              tx=mx-(getSize().width-width)/2;
              ty=my-(getSize().height-height)/2;
              if(tx<0 || ty<0 || tx>width-1 || ty>height-1)
                  pix=0;
              else
                  pix=pixels[ty*width+tx];
              Color pixColor= new Color(pix);
              colorBox.setBackground(pixColor);
              tmp=pix & 0xFF;
              blue_l.setText(String.valueOf(tmp));
              tmp=pix>>8 & 0xFF;
              green_l.setText(String.valueOf(tmp));
              tmp=pix>>16 & 0xFF;
              red_l.setText(String.valueOf(tmp));
              col=0;
              row=0;
            }
        else if(data==2) // New col
            {
              col+=2;
              row=0;
            }

        else if(data==3) // End frame
            {
              outWin.append("Dump frame Complete" );
               source.newPixels(0,0,width,height);
              start=1;
              row=0;
              col=0;
              return 0;
            }
        else
            {  // A pixel!
              int red,green,blue,green2;
              red=data;
              green=mySerial.readByte();
              blue=mySerial.readByte();
              green2=green;
              pix=0;
              pix+=blue;
              pix+=green<<8;
```

```
                pix+=red<<16;

                Color pixColor= new Color(pix);
                pix=0;
                pix+=blue;
                pix+=green2<<8;
                pix+=red<<16;
                Color pixColor2= new Color(pix);
                if(row<height && col<width)
                    {
                pixels[row*width+col] = pixColor.getRGB();
                pixels[row*width+col+1] = pixColor2.getRGB();
                row++;
                    }
              }
          data=mySerial.readNonBlock();
          }
  }
  catch(Exception e){
      System.out.println(e);
  }


 source.newPixels(0,0,width,height);
 return 1;
}

/*
  The buffer that holds the background image.
  The lines for line mode on get mean are drawn
  into the background buffer too.
*/
public void update(Graphics g)
{
  if(offScreenImage==null)
      {
        offScreenImage=createImage(getSize().width,getSize().height);
        offScreenGC=offScreenImage.getGraphics();
      }
  paint(offScreenGC);
  g.drawImage(offScreenImage,0,0,this);
}
/*
  This is where the bitmap and tracking boxes are drawn.
  The actual background is add in update()
*/
public void paint(Graphics g)
{

  int w_width,w_height;
  w_width=getSize().width;

  w_height=getSize().height;
  int sx=mySW.rX();
  int sy=mySW.rY();
  g.setColor(Color.black);
  g.fillRect(0,0,w_width,w_height);
  g.drawImage(image,(w_width-width)/2+sx,(w_height-height)/2+sy,this);

  // Tracking box
```

```
      if(xcoord!=width && ycoord!=height && x2coord!=width &&
y2coord!=height && zcoord!=0)
        {
            if(conf>50) g.setColor(Color.green);
            else
            if(conf>0) g.setColor(Color.yellow);
            else g.setColor(Color.red);
            g.fillRect(x2coord+(w_width-width)/2,y2coord+(w_height-
height)/2, xcoord-x2coord,ycoord-y2coord );
        }

      // Line mode for bitmap drawing
      if(lm==1 && tc==1)
        {
            g.setColor(Color.cyan);
            int bx,by;
            for(int k=0; k<lmIndex; k++ )

              for(int j=0; j<10; j++ )
                  {

                    for(int l=0; l<8; l++ )
                        {
                          bx=w_width-(w_width-width)/2-((j*8+l)*2);
                          by=(w_height-height)/2+k*3;
                          if( (bmask[j][k]<<l & 128)==128)
                              g.fillRect(bx,by, 2, 3 );
                        }

                  }
        }

      // Red dot for middle mass mode
      if(mm==1)
        {
            if(mmx!=0 || mmy!=0)
              {
                  g.setColor(Color.red);
                  g.fillRect((width-mmx*2)+(w_width/2)-width/2,(144-
mmy)+(w_height/2)-height/2, 5, 5 );
              }
        }

    }



}
```

CamSetting.java

```
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;

public class camSettings extends Canvas
{
    Frame camSettings_f;
```

```
Panel camSettings_p;
Choice fps_c,cam_c,gain_c;

camSettings()
{
  camSettings_f= new Frame("Registres de la Camera");
  Panel camSettings_p = new Panel();
  camSettings_f.setBackground(Color.lightGray);
  camSettings_f.setLayout(new GridLayout(3,2));
  camSettings_f.add(new Label("Clock (fps)"));
  fps_c = new Choice();
  fps_c.addItem("17");
  fps_c.addItem("13");
  fps_c.addItem("11");
  fps_c.addItem("9");
  fps_c.addItem("8");
  fps_c.addItem("7");
  fps_c.addItem("6");
  fps_c.addItem("5");
  fps_c.addItem("4");
  camSettings_f.add(fps_c);
  camSettings_f.add(new Label("Mode de l'Image"));
  cam_c = new Choice();
  cam_c.addItem("RGB WB Off");
  cam_c.addItem("RGB WB On");
  cam_c.addItem("YUV WB Off");
  cam_c.addItem("YUV WB On");
  camSettings_f.add(cam_c);
  camSettings_f.add(new Label("Gain Automatique"));
  gain_c = new Choice();
  gain_c.addItem("On");
  gain_c.addItem("Off");
  camSettings_f.add(gain_c);
  camSettings_f.setSize(250,100);
  camSettings_f.setLocation(0,480);
  camSettings_f.show();

}
public void hide() { camSettings_f.hide(); }
public void show() { camSettings_f.show(); }

public String getString()
{
  int color_mode=44;
  int fps=2;
  int gain=33;

  switch(cam_c.getSelectedIndex())
      {
      case 0: color_mode=40; break;
      case 1: color_mode=44; break;
      case 2: color_mode=32; break;
      case 3: color_mode=36; break;
      }
  switch(fps_c.getSelectedIndex())
      {
      case 0: fps=2; break;
      case 1: fps=3; break;
      case 3: fps=4; break;
      case 4: fps=5; break;
      case 5: fps=6; break;
```

```
            case 6: fps=7; break;
            case 7: fps=8; break;
            case 8: fps=9; break;
            case 9: fps=10; break;
            case 10: fps=11; break;
            case 11: fps=12; break;
            }
        switch(gain_c.getSelectedIndex())
            {
            case 0: gain=33; break;
            case 1: gain=32; break;
            }
        return( "CR 18 " + new Integer(color_mode).toString()
                + " 17 " + new Integer(fps).toString()
                + " 19 " + new Integer(gain).toString());
    }
}
```

ChannelWindow.java

```
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;

public class channelWindow extends Canvas
{

    int w,h;
    int chan1[];
    int update;

    channelWindow(int width,int height)
    {
      w=width;
      h=height;
      chan1=new int[width*height];
      update=0;
    }

    public void splitChannels(int pixels[])
    {
      chan1=pixels;
      update=1;
      repaint();
    }

    public void update(Graphics g)
    {

      paint(g);

    }

    /*
       This is where most of the work is done.
       The image gets read and drawn directly to
       the screen.
    */
```

```
    public void paint(Graphics g)
    {
      int mainPix,x,y,ch1,ch2,ch3,tmp;

      if(update==1)
      for(x=0; x<w; x++ )
          for(y=0; y<h; y++ )
            {
                mainPix=chan1[y*w+x];
                tmp=mainPix & 0xFF0000;
                tmp=tmp>>16;
                ch3=tmp+(tmp<<8)+(tmp<<16);
                Color pixColor3= new Color(ch3);
                g.setColor(pixColor3);
                g.fillRect(x+10,y+10,1,1);


                tmp=mainPix & 0xFF00;
                tmp=tmp>>8;
                ch2=tmp+(tmp<<8)+(tmp<<16);
                Color pixColor2= new Color(ch2);
                g.setColor(pixColor2);
                g.fillRect(x+10,y+h+20,1,1);

                tmp=mainPix & 0xFF;
                ch1=tmp+(tmp<<8)+(tmp<<16);
                Color pixColor= new Color(ch1);
                g.setColor(pixColor);
                g.fillRect(x+10,y+(2*h)+30,1,1);

            }
      update=0;
    }
}
```

ColorTrack.java

```
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;

public class colorTrack extends Canvas
{
Frame tracker_f;
Panel tracker_p;
TextField rMin,rMax,gMin,gMax,bMin,bMax;


    colorTrack()
    {
      tracker_f= new Frame("Suivi de couleur");
      Panel tracker_p = new Panel();
      tracker_f.setBackground(Color.lightGray);
      tracker_f.setLayout(new GridLayout(3,3));
      rMin=new TextField("0");
      rMax=new TextField("0");
      gMin=new TextField("0");
      gMax=new TextField("0");
```

```
      bMin=new TextField("0");
      bMax=new TextField("0");
      tracker_f.add(new Label("Red (min max)"));
      tracker_f.add(rMin);
      tracker_f.add(rMax);
      tracker_f.add(new Label("Green (min max)"));
      tracker_f.add(gMin);
      tracker_f.add(gMax);
      tracker_f.add(new Label("Blue (min max)"));
      tracker_f.add(bMin);
      tracker_f.add(bMax);
      tracker_f.setSize(300,120);
      tracker_f.setLocation(260,100);
      tracker_f.show();
    }
    public void hide() { tracker_f.hide(); }
    public void show() { tracker_f.show(); }

    public String sendString()
    {
      return("TC " +rMin.getText()+" "+rMax.getText()+" "+gMin.getText()
               + " " +gMax.getText()+" "+bMin.getText()+" "+bMax.getText()
              + "\r" );
    }
}
```

CommWindow.java

```
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;


public class commWindow extends Canvas
{
    Frame comm_f;
    Panel comm_p;
    TextField comm_t;
    Dialog comm_d;
    int done;

    commWindow()
    {
      done=0;
      comm_f = new Frame();
      comm_p= new Panel();
      String tempOs = new String( System.getProperty("os.name"));
      if(tempOs.startsWith("Windows"))
          comm_t = new TextField("1");
      else
          comm_t = new TextField("/dev/ttyS0");
      Button comm_b = new Button("Ok");
      comm_b.addActionListener( new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                done=1;
```

```
            }
          } );
      comm_d = new Dialog(comm_f,"Sélection du port série");
      comm_p.add(comm_t);
      comm_p.add(comm_b);
      comm_d.add(comm_p);
      comm_d.setBounds(50,50,180,70);
      comm_d.show();

    }


    public String getPort()
    {
      return comm_t.getText();
    }
    public int ready()
    {
      if(done==0) return 0;
      comm_d.setVisible(false);
      return 1;
    }



}
```

MainWindow.java

```
//programme allégé de CMUcamGUI propre à notre utilité

   import java.awt.*;
   import java.lang.*;
   import java.awt.event.*;
   import java.util.EventListener;
   import java.io.*;

   public class mainWindow implements ActionListener, ItemListener
   {
      volatile int commandVal;
      Button sendstring_b;
      TextField rawCommand_f;
      Thread updateThread;
      CameraImage cImage;
      mainWindow()
      {
         commandVal=0;
         commWindow cWindow = new commWindow();  // Serial Config Window
         while(cWindow.ready()==0);
         Frame main_f = new Frame("cam");
         cImage = new CameraImage(160,144,cWindow.getPort()); // Set serial
port
         rawCommand_f=new TextField("gv");
         sendstring_b = new Button("Send String");
         sendstring_b.addActionListener(this);

      // Make menus and such

         Panel myPanel = new Panel();
         MenuBar menuBar = new MenuBar();
         Menu file_m = new Menu("File");
```

```
        file_m.add(makeMenuItem("Partir"));
        Menu commands_m = new Menu("Commandes",true);

        commands_m.add(makeMenuItem("RAZ camera"));
        commands_m.add(makeMenuItem("Suivi de couleur"));
        commands_m.add(makeMenuItem("Suivi de fenêtre"));
        commands_m.add(makeMenuItem("Calibrage fenêtre"));
        commands_m.add(makeMenuItem("Callibrage du registre de la
camera"));


        menuBar.add(file_m);
        menuBar.add(commands_m);
        main_f.setMenuBar(menuBar);
        main_f.setSize(230,250);
        main_f.add("Center",cImage);
        myPanel.setLayout( new BorderLayout());

        myPanel.add( "Center",rawCommand_f);
        myPanel.add( "East", sendstring_b);
        main_f.add("South",myPanel);


        main_f.show();
        updateThread=new Thread(
                                new Runnable() {
                                   public void run() {
                                      grabFrame();}
                                });
        updateThread.start();
        cImage.writeText("Recherche de la camera...");
        boolean conf;
        conf=cImage.sendCommand("rs\r");
        clear();
        if(!conf)  // Try one more time if it failed
        {
           cImage.sendCommand("rs\r");
           clear();
        }
        commandVal=0;
     }
     private void grabFrame()
     {
        int tempVal=-1,cnt;
        while(true)
        {
           try{
              switch(commandVal)
              {

                 case 2: // Suivi de couleur
                    tempVal=cImage.trackColor(0);
                    break;
                 case 3:
                    commandVal=0;
                    cnt=0;
                    while(!cImage.idle());
                    break;
                 case 4:
                    commandVal=0;
                    cImage.sendCommand(rawCommand_f.getText()+'\r');
```

```
                break;
            case 5: // Suivi de fenêtre
                tempVal=cImage.trackColor(1);
                break;
            case 6: // Suivi de couleur
                tempVal=cImage.getMean(0);
                break;
            case 7: // Calibrage fenêtre
                commandVal=0;
                tempVal=cImage.sw();
                break;
            case 8: //Callibrage du registre de la camera
                commandVal=0;
                cImage.setCamera();
                break;
            default:
         }
         if(tempVal==0){ commandVal=0; clear(); tempVal=-1; }
         Thread.sleep(1);
      }
         catch(InterruptedException e) {
         }
   }
}
public void actionPerformed(ActionEvent e)
{
   String command = e.getActionCommand();
   if( command.equals("Partir"))
      System.exit(0);

   if( command.equals("Suivi de couleur"))
   {
      clear();
      commandVal=2;
   }
   if( command.equals("Suivi de fenêtre"))
   {
      clear();
      commandVal=5;
   }
   if( command.equals("Calibrage fenêtre"))
   {
      clear();
      commandVal=7;
   }
   if( command.equals("Callibrage du registre de la camera"))
   {
      clear();
      commandVal=8;
   }
   if( command.equals("RAZ camera"))
   {
      boolean conf;
      conf=cImage.sendCommand("rs\r");
      clear();
      if(!conf)
      {
         cImage.sendCommand("rs\r");
         clear();

      }
```

```
            commandVal=0;
        }
        if( command.equals("Send String"))
        {
            commandVal=0;
            while(!cImage.idle());
            commandVal=4;
        }
    }

    private void clear()

    {
        int cnt;
        commandVal=0;

        for(cnt=0; cnt<100; cnt++ )
        {
            try{
                Thread.sleep(500);
            }
                catch(Exception blah){
                }
            if(cImage.idle())
                break;
        }
        if(cnt==99) System.out.println("<clear> time out" );
        cImage.flushBuf();
    }

/*
This is for check box menus
*/
    public void itemStateChanged(ItemEvent e)
    {
        String item = e.getItem().toString();
        int action = e.getStateChange();
        if( item.equals("Color Picker"))
        {
            if(action==2)cImage.picker_f.hide();
            else cImage.picker_f.show();
        }

        if( item.equals("Suivi de couleur"))
        {
            if(action==2)cImage.hideTrack();
            else cImage.showTrack();
        }

    }

    private MenuItem makeMenuItem( String name )
    {
        MenuItem m = new MenuItem(name);
        m.addActionListener(this);
        return m;

    }

    private CheckboxMenuItem makeCheckMenuItem( String name,int val )
    {
```

```
        CheckboxMenuItem m = new CheckboxMenuItem(name);
        if(val==1)
           m.setState(true);
        else
           m.setState(false);
        m.addItemListener(this);
        return m;

     }
  }
```

MeanWindow.java

```
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;


public class meanWindow extends Canvas
{
Frame mean_f;
Panel mean_p,cmin,cmean,cmax;
TextField rMean,rDev,gMean,gDev,bMean,bDev;


    meanWindow()
    {
      mean_f= new Frame("Mean Color Data");
      Panel tracker_p = new Panel();
      mean_f.setBackground(Color.lightGray);
      mean_f.setLayout(new GridLayout(6,3));
      mean_f.add(new Label(" "));
      mean_f.add(new Label("Mean"));
            mean_f.add(new Label("Deviation"));
      mean_f.add(new Label("Rouge"));
      rMean=new TextField("?");
      rDev=new TextField("?");
      cmin=new Panel();
      mean_f.add(rMean);
      mean_f.add(rDev);

      mean_f.add(new Label("Vert"));
      gMean=new TextField("?");
      gDev=new TextField("?");
      cmean=new Panel();
      mean_f.add(gMean);
      mean_f.add(gDev);

      mean_f.add(new Label("Bleu"));
      bMean=new TextField("?");
      bDev=new TextField("?");
      cmax=new Panel();
      mean_f.add(bMean);
      mean_f.add(bDev);

      mean_f.add(new Label("low"));
```

```
      mean_f.add(new Label("mean"));
      mean_f.add(new Label("high"));
      mean_f.add(cmin);
      mean_f.add(cmean);
      mean_f.add(cmax);

      mean_f.setSize(250,200);
      mean_f.setLocation(0,270);
      mean_f.show();
    }
    public void hide() { mean_f.hide(); }
    public void show() { mean_f.show(); }

    public void update(int rm,int gm,int bm,int rd,int gd,int bd)
    {
      int r,g,b;
      rMean.setText((new Integer(rm)).toString());
      gMean.setText((new Integer(gm)).toString());
      bMean.setText((new Integer(bm)).toString());
      rDev.setText((new Integer(rd)).toString());
      gDev.setText((new Integer(gd)).toString());
      bDev.setText((new Integer(bd)).toString());

      r=rm-rd;
      g=gm-gd;
      b=bm-bd;
      if(r>254) r=254; if(r<0)r=0;

      if(g>254) g=254; if(g<0)g=0;
      if(b>254) b=254; if(b<0)r=0;
      cmin.setBackground(new Color(r,g,b));
      cmean.setBackground(new Color(rm,gm,bm));
      r=rm+rd;
      g=gm+gd;
      b=bm+bd;
      if(r>254) r=254; if(r<0)r=0;
      if(g>254) g=254; if(g<0)g=0;
      if(b>254) b=254; if(b<0)r=0;
      cmax.setBackground(new Color(r,g,b));
    }

}
```

OutWindow.java

```
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;


public class outWindow extends Canvas
{
Frame out_f;
Panel out_p;
TextArea out_t;
int count;
```

```
    outWindow()
    {
      out_f= new Frame("Sortie de fenêtre");
      Panel out_p = new Panel();
      out_f.setBackground(Color.lightGray);
      out_t=new TextArea("");
      out_f.add(out_t);
      out_f.setSize(400,300);
      out_f.setLocation(260,250);
      out_f.show();
      int count=0;
    }

    public void append(String data)
    {
      out_t.append( data + "\n");
      count++;
      if(count>1000)
      {
       out_t.setText( data + "\n");
      count=0;
      }
    }
}
```

RawWindow.java

```
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;


public class rawWindow extends Canvas
{
Frame raw_f;
Panel raw_p;
TextArea raw_t;


    rawWindow()
    {

      raw_f= new Frame("Raw Window");
      Panel raw_p = new Panel();
      raw_f.setBackground(Color.lightGray);
      raw_t=new TextArea("");
      raw_f.add(raw_t);
      raw_f.setSize(400,300);
      raw_f.setLocation(400,250);
      raw_f.show();
    }

    public void append(String data)
    {
      raw_t.append( data );
```

```
    }
 public void nl()
    {
      raw_t.append( "\n");
    }
}
```

SerialComm.java

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\
 * This file is part of CMUcamGUI, a java program that helps     *
 * interface with the CMUcam Vision Board.                       *
 * Contact cmucam@cs.cmu.edu, or see                             *
 * http://www.cs.cmu.edu/~cmucam for more information.           *
 *                                                               *
 * Copyright 2001 Anthony Rowe                                   *
 *                                                               *
 * This program is free software; you can redistribute it and/or *
 * modify it under the terms of the GNU General Public License   *
 * as published by the Free Software Foundation - version 2.     *
 *                                                               *
 * This program is distributed in the hope that it will be       *
 * useful, but WITHOUT ANY WARRANTY; without even the implied    *
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR       *
 * PURPOSE.  See the GNU General Public License in file COPYING  *
 * for more details.                                             *
 *                                                               *
 * You should have received a copy of the GNU General Public     *
 * License along with this program; if not, write to the Free    *
 * Software Foundation, Inc., 59 Temple Place - Suite 330,       *
 * Boston, MA 02111, USA.                                        *
\* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;

/*
This file was a first attempt at making a cross platform
program.  This may seem strange, but this program switches
serial functions based on the OS.  Eventually I might get
around to using the java serial calls, but for now this
seems to work.
*/

public class serialComm
{
    FileOutputStream sPortOut;
    FileInputStream sPortIn;
    int os;
    serialComm(String commPort)
    {
      String tempOs = new String( System.getProperty("os.name"));
      if(tempOs.startsWith("Windows")) os=0;
         else
         os=1;
      if(os==0)
          {//Windows
            System.out.println( "Windows Detected" );
```

```
             int error;
            error = serialPort.openSerial(new Integer(commPort).intValue(),
5);

            if (error != 0) {
                  System.err.println("Error " + error + " during
openSerial()");
            System.exit(0);
            }
            serialPort.setReadTimeout(500); //Illah - make this much
smaller?

            System.out.println("serial port successfully opened!");

          }
      else
          { // Unix Serial Init
            System.out.println( "Linux Detected" );
            try{
                sPortIn = new FileInputStream( commPort );
                sPortOut = new FileOutputStream( commPort );
            }
            catch(Exception e){ System.out.println(e);}
          }
    }
    /*
      Reads in an ascii number and returns an int.
      This function blocks.
    */
    public int readNum()
    {
      char one,two,three,four;
      one=readByte();
      two=readByte();
      if( two==' '|| two=='\r' )
          return((int)(one-'0'));
      three=readByte();
      if( three==' ' || three=='\r' )
          return((int)(one-'0')*10+(two-'0'));
      four=readByte();
      return((int)(one-'0')*100+(two-'0')*10+(three-'0'));

    }

    public char readByte()
    {
       Date time = new Date();
       long wdTimer=time.getTime();
      if(os==0)
          {//Windows
            int val=-1;
            while(val==-1)
                  {
                  val = serialPort.readByte();
                  Date time2 = new Date();
                      if(time2.getTime()-wdTimer>1000)
                        {
                            System.out.println("<Serial Read> time out");
                            return 0;
                        }
```

```
            }
        return((char)val);
      }
  else
      { // Unix Serial Init
        try{
            while(sPortIn.available()==0 )
              {
                  Date time2 = new Date();
                  if(time2.getTime()-wdTimer>1000)
                    {
                        System.out.println("<Serial Read> time out");
                        return 0;
                    }
              }
            return((char)sPortIn.read());
        } catch(Exception e){System.out.println(e); }
      }
  return 0;
}

public char readNonBlock()
{
  if(os==0)
      {//Windows
        int val;
        val = serialPort.readByte();
        if(val==-1) val=0;
        return((char)val);
      }
  else
      { // Unix Serial Init
        try{
            if(sPortIn.available()==0 ) {
              return 0;
            }
            return((char)sPortIn.read());
        } catch(Exception e){System.out.println(e); }
      }
  return 0;
}


public void writeStr(String in)
{
  if(os==0)
      {//Windows
        int i;
        byte[] bytes = in.getBytes();
        for (i = 0; i < bytes.length; i++) {
            int error = serialPort.sendByte(bytes[i]);
            if (error != 0) {
              System.out.println( "Serial Send error" );
              break;
            }
        }

      }
  else
      { // Unix Serial Init
        try{
```

```
                sPortOut.write(in.getBytes());
              } catch(Exception e){System.out.println(e); }
            }
        }
}
```

SerialConfig.c

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\
 * This file is serialConfig, a C program that sets your serial  *
 * port to 115,200 baud 8n1                                      *
 * Contact cmucam@cs.cmu.edu, or see                             *
 * http://www.cs.cmu.edu/~cmucam for more information.           *
 *                                                               *
 * Copyright 2001 Anthony Rowe                                   *
 *                                                               *
 * This program is free software; you can redistribute it and/or *
 * modify it under the terms of the GNU General Public License   *
 * as published by the Free Software Foundation - version 2.     *
 *                                                               *
 * This program is distributed in the hope that it will be       *
 * useful, but WITHOUT ANY WARRANTY; without even the implied    *
 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR       *
 * PURPOSE.  See the GNU General Public License in file COPYING  *
 * for more details.                                             *
 *                                                               *
 * You should have received a copy of the GNU General Public     *
 * License along with this program; if not, write to the Free    *
 * Software Foundation, Inc., 59 Temple Place - Suite 330,       *
 * Boston, MA 02111, USA.                                        *
\* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <stdio.h>
#include <math.h>
#include <termios.h>
#include <unistd.h>
#include <png.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <linux/types.h>
#include <linux/videodev.h>
#include <sys/ioctl.h>
#include <sys/signal.h>


int main(int argc, char *argv[]) {
  struct termios oldtio, newtio;
  struct sigaction saio;
  int s_port;
  char port_name[32];
  // Serial Setup

if(argc<=1)
      {
      printf("No command line port entered, using default\n" );
      strcpy(port_name,"/dev/ttyS0" );
      s_port=open(port_name, O_RDWR | O_NOCTTY | O_NONBLOCK);
      }
```

```
     else
      {
      strcpy(port_name,argv[1]);
      s_port=open(port_name,O_RDWR | O_NOCTTY | O_NONBLOCK);
      }
  if (s_port<0) {
    printf("error opening serial port\n");
    exit(-1);
  }


  tcgetattr(s_port, &oldtio);
  newtio=oldtio;
  newtio.c_cflag=0x18b2;
  newtio.c_lflag=0x0;
  newtio.c_iflag=0x1;
  newtio.c_oflag=0x0;
  newtio.c_ispeed=0xbffffd08;
  newtio.c_ospeed=0x40036e78;
  newtio.c_line=0x0;
  tcsetattr(s_port,TCSAFLUSH,&newtio);
  printf( "%s Set to 115,200 Baud 8n1\n",port_name,argc);
}
```

SerialPort.java

```
import java.lang.*;
public class serialPort extends java.lang.Object
{
    // open serial port at specified baud with other appropriate
parameters.
    // returns:
    // 0 on success
    // -1 on failure to open serial port
    // -2 on failure to read port state
    // -3 on failure to set port state
    // coms are: 1 = "COM1"; 2 = "COM2" et cetera
    // baud rates are: 1 = 9600; 2=19200; 3=38400; 4=57600; 5=115200;
6=230400
    public synchronized native static int openSerial(int comNum, int
baudSpec);
    public synchronized native static int closeSerial();
    // returns 0 on success or -1 on failure to send byte
    public synchronized native static int sendByte(int theByte);
    // initialTimeout is measured in milliseconds //
    // returns -6 on failure to set timeout, 0 on success
    public synchronized native static int setReadTimeout(int
initialTimeout);
    // returns -1 in error or timeout, else returns int between 0 and 255
    public synchronized native static int readByte();

   static {
    System.loadLibrary("sserial");
   }

}
```

SetWindow.java

```
import java.awt.*;
```

```
import java.awt.image.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Object;
import java.util.*;


public class setWindow extends Canvas
{
    Frame setWindow_f;
    Panel setWindow_p;
    TextField x1_t,y1_t,x2_t,y2_t;
    int x1,x2,y1,y2;


    setWindow()
    {
      setWindow_f= new Frame("Taille de la fenêtre");
      Panel setWindow_p = new Panel();
      setWindow_f.setBackground(Color.lightGray);
      setWindow_f.setLayout(new GridLayout(2,3));
      x1_t=new TextField("1");
      y1_t=new TextField("1");
      x1=1;
      y1=1;
      x2_t=new TextField("80");
      y2_t=new TextField("143");
      setWindow_f.add(new Label("Upper Left (x,y)"));
      setWindow_f.add(x1_t);
      setWindow_f.add(y1_t);
      setWindow_f.add(new Label("Lower Right(x2,y2)"));
      setWindow_f.add(x2_t);
      setWindow_f.add(y2_t);
      setWindow_f.setSize(300,80);
      setWindow_f.setLocation(0,590);
      setWindow_f.show();

    }
    public void hide() { setWindow_f.hide(); }
    public void show() { setWindow_f.show(); }

    public String sendString()
    {
      x1=(new Integer(x1_t.getText())).intValue();
      y1=(new Integer(y1_t.getText())).intValue();

      return("SW " +x1_t.getText()+" "+y1_t.getText()+" "+x2_t.getText()
               + " " +y2_t.getText()+ "\r" );


    }

    public int rX(){ return x1; }
    public int rY(){ return y1; }

}
```

ZoomBox.java

```
import java.awt.*;
import java.awt.image.*;
```

```
import java.awt.event.*;

public class zoomBox extends Canvas
{
Color pixColor;
zoomBox()
{


}

public void updateBox(int x,int y,int myColor)
{
pixColor = new Color(myColor);
repaint();
}

public void update(Graphics g)
{
paint(g);
}
public void paint(Graphics g)
{
g.setColor(pixColor);
g.fillRect(0,0,getSize().width,getSize().height);
}

}
```