REFERENCES

1. Wu, C. T. *An Introduction to Object-Oriented Programming with Java*. 2nd ed. New York: McGraw-Hill, 2001. p. 1.
2. Sato, T., Kajiwara, K., Hoshino, H. and Nakamura, T. Attempt of Numerical Expression of Color Emotion. *Use of Color from fundamentals to Applications for the 21st Century*, 1999: pp. 65-70.
3. Freund, R. J. and Wilson, W. J. *Statistical Methods*, Academic. San Diego: 1993. pp 262-264.
4. Chase, W. and Bown, F. *General Statistics*. 2nd ed. Singapore: John Wiley & Sons, 1992: pp. 479-483.
5. Berns, R. S. *Principles of Color Technology*. 3rd ed. New York: John Wiley & Sons, 2000.
6. Field, G. G. *Color and Its Reproduction*. Pennsylvania: Graphic Arts Technical Foundation, 1988. p. 65.
7. Hunt, R. W. G. *Measuring Color*. 2nd ed. Chichester: Ellis Hourwood, 1991. p. 140.
8. Minolta, comp. *Precise Color Communication*. Osaka: Minolta, 1994. p. 46.
9. Kang, H. R. *Color Technology for Electronic Imaging Devices*. Washington: SPIE-The International Society for Optical Engineering, 1997. pp. 342-348.
10. Berns, R. S., Motta, R. J. and Gorzynski, M. E. CRT colorimetry. Part I: Theory and practice. *Color Res. Applic.*, **4**, 1970. pp. 871-874.
11. Dalal, E. N., Blaszak, S. E. and Swanton, P. C. Projection efficiency of color electrophotographic transparencies. *Proc. Fifth International Congress on Adv. In Non-Impact Printing Technologies*., 1989. pp. 192-195.
12. Dalal, E. N., Paine, A. J., Blaszak, S. E. and Morrison, I. D. Predictive modeling of color projection quality: II. Experimental validation. *Proc. IS&T 47th Annual Conference/ICPS*., 1994. pp. 763-766.
13. Bangchokdee, Y. Master Degree Thesis, Chulalongkorn University, 2000.
14. Wu, C. T. *An Introduction to Object-Oriented Programming with Java*. 2nd ed. New York: McGraw-Hill, 2001. pp. 11-16.

15. Nakamura, T., Hoshino, H., Sato, T. and Kajiwara, K. Numerical Evaluation of Color Image Words on Colorimetry. *AIC Color 97*, **2**, 1997. pp. 699-702.

16. Ho, W. T. Master Degree Thesis, University of Leeds., 1996.

17. Ngampatipatpong, D. Master Degree Thesis, Chulalongkorn University, 1999.

18. Sato, T., Kajiwara, K., Hoshino, H. and Nakamura, T. Quantitative Assessment of Light-Dark, Deep-Pale and Heavy-Light Color Images. *The Journal of the society of Fiber Science and Technology*, **53**, 1997. p. 7.

19. Sato, T., Kajiwara, K., Hoshino, H. and Nakamura, T. The Attempt of Quantitative Assessment for Vivid-Sombre, Gaudy-Plain, Striking-Subdued, Dynamic-Passive, Distinct-Vague, Transparent-Turbid, Soft-Hard and Strong-Weak Color Images. *The Journal of the society of Fiber Science and Technology*, **53**, 1997. p. 131.

20. Ngampatipatpong, D., Hansuebsai, A., Pungrassamee, P. and Sato, T. Color Emotion Scales for Thais. *Use of Color from Fundamentals to Applications for the 21st Century*, 1999. pp. 71-76.

21. Sato, T., Kawahito, Y., Kajiwara, K., Hoshino, H. and Nakamura, T. Attempt of Numerical Expression of Color Emotion. *Use of Color from Fundamentals to Applications for the 21st Century*, 1999. pp. 65-70.

22. Xin, J. H. and Cheng, K-M. Quantitative Evaluation of Colour Emotion. *Microsymposium on Colour Research and Application*, 2000. pp. 71-86.

23. Berns, R. S., Motta, R. J. and Gorzynski, M. E. CRT colorimetry. Part I: Theory and practice. *Color Res. Applic.*, **18**, 1993. pp. 299-314.

APPENDICES

APPENDIX A

Program 01

Java application program

```
import java.io.*;
import java.util.*;

public class GlobalVar
{
	public static String MONITOR_FILE = "mymonitor.txt";

	//public static boolean g_datavalid;
	public static Boolean g_datavalid = new Boolean(false); // my var

	public static double g_coefxyz2rgb[][]= new double[3][3];
	public static double g_coefrgb2xyz[][]= new double[3][3];

	public static double g_gog[][] = new double[3][2]; //[3][2] GOG modal
	public static double g_sy_max[] = new double[8]:
	public static double g_sx_max[] = new double[8];

	public static double g_xmin, g_ymin, g_zmin;
	public static double g_xyzmin[] = { 0.0, 0.0, 0.0 }; //my var


	public static void calibrate(){
		calibrate(g_coefxyz2rgb, g_coefrgb2xyz,
			g_gog, g_sx_max, g_sy_max,
			g_xyzmin, g_datavalid);
		System.out.println( " File name = " + MONITOR_FILE );
		//for( int i=0; i<3; i++ )
			//for( int j=0; j<3; j++ )
				//System.out.println( g_coefrgb2xyz[i][j]);
		for( int i=0;i<3;i++ )
			for(int j=0;j<2;j++)
				System.out.println( g_gog[i][j]);
	}

	public static void calibrate(double coefxyzrgb[][], double coefrgbxyz[][],
			double gog[][], double sx_max[], double sy_max[],
//			double& xmin, double& ymin, double& zmin, bool& valid)
			double xyzmin[], Boolean valid)
	{
		//double sy[18],sx[18],lmax[8],l[18];
		//double xl[3][18],x0,y0,z0;
		double sy[] = new double[18];
		double sx[] = new double[18];
		double lmax[] = new double[8];
		double l[] = new double[18];
		double xl[][] = new double[3][18];
		double x0,y0,z0;
		double sx_min,sy_min,lmin;
			sx_min = sy_min = lmin = 0;
		//int i,j,ns,nt,dac[3][18],dummy;
		int i,j,ns,nt,dummy;
		int dac[][] = new int[3][18];
		int inter,neq;

		String oneLine;

		valid = new  Boolean( false );
```

```
                ns = 15;
                nt = 40;
                inter = 1;
                neq = 1;

                //using greys + 1 white.
                ns = ns +1;

                //FILE *fp;
                try{
                        //DataInputStream fp = new DataInputStream( new
BufferedInputStream ( new FileInputStream (MONITOR_FILE)));
                        BufferedReader fp = new BufferedReader( new FileReader(
MONITOR_FILE ));
                        //valid = false;

                        //check the existance of the file monitor.h
                        //if(_access(MONITOR_FILE, 00) != 0)
                        //      return ;

                        //fp = fopen(MONITOR_FILE,"r");

                        //reading white, r,g,b maximum
                        for (i=0;i<4;i++){
                                //fscanf(fp,"%d %d %d %lf %lf %lf",&dummy,&dummy,
&dummy,
                                //                      &sx_max[i],&sy_max[i],&lmax[i]);
                                oneLine = fp.readLine();
                                StringTokenizer token = new StringTokenizer( oneLine );
                                int col = 0;
                                while( token.hasMoreTokens() ){
                                        switch( col ){
                                                case 3 : sx_max[i] = Double.parseDouble(
token.nextToken() ); break;
                                                case 4 : sy_max[i] = Double.parseDouble(
token.nextToken() ); break;
                                                case 5 : lmax[i] = Double.parseDouble(
token.nextToken() ); break;
                                                default : token.nextToken();
                                        }
                                        col++;
                                }
                        }

                        //find X0,Y0,Z0
                        x0 = 100.0/sy_max[0]*sx_max[0];
                        y0 = 100.0;
                        z0 = 100.0/sy_max[0]*(1.0-sx_max[0]-sy_max[0]);

                        //reading grey samples
                        for (i=0;i<ns;i++) {
                                //fscanf(fp,"%d %d %d %lf %lf %lf",&dac[0][i],&dac[1][i],
                                //                      &dac[2][i],&sx[i],&sy[i],&l[i]);
                                oneLine = fp.readLine();
                                StringTokenizer token = new StringTokenizer( oneLine );
                                int col = 0;
                                while( token.hasMoreTokens() ){
                                        switch( col ){
```

```
                                        case 0 : dac[0][i] = Integer.parseInt(
token.nextToken() ); break;
                                        case 1 : dac[1][i] = Integer.parseInt(
token.nextToken() ); break;
                                        case 2 : dac[2][i] = Integer.parseInt(
token.nextToken() ); break;
                                        case 3 : sx[i] = Double.parseDouble(
token.nextToken() ); break;
                                        case 4 : sy[i] = Double.parseDouble(
token.nextToken() ); break;
                                        case 5 : l[i] = Double.parseDouble(
token.nextToken() ); break;
                                        default : token.nextToken();
                                }
                                col++;
                        }

                        if(i==ns-1)
                        {
                                lmax[4]=l[i];
                                sx_max[4]=sx[i];
                                sy_max[4]=sy[i];
                        }
                }
                for (i=5;i<9;i++){
                        //fscanf(fp,"%d %d %d %lf %lf %lf",&dummy,&dummy,
&dummy,
                        //              &sx_max[i],&sy_max[i],&lmax[i]);
                        oneLine = fp.readLine();
                        StringTokenizer token = new StringTokenizer( oneLine );
                        int col = 0;
                        while( token.hasMoreTokens() ){
                                if (i==8){
                                        //fscanf(fp,"%d %d %d %lf %lf %lf",&dummy,
&dummy,&dummy,
                                        //                      &sx_min,&sy_min,
&lmin);   // dac 0 0 0 point
                                        switch( col ){
                                                case 3 : sx_min =
Double.parseDouble( token.nextToken() ); break;
                                                case 4 : sy_min =
Double.parseDouble( token.nextToken() ); break;
                                                case 5 : lmin = Double.parseDouble
( token.nextToken() ); break;
                                                default : token.nextToken();
                                        }

                                } else {
                                        switch( col ){
                                                case 3 : sx_max[i] =
Double.parseDouble( token.nextToken() ); break;
                                                case 4 : sy_max[i] =
Double.parseDouble( token.nextToken() ); break;
                                                case 5 : lmax[i] =
Double.parseDouble( token.nextToken() ); break;
                                                default : token.nextToken();
                                        }
                                }
```

```
                                col++;
                        }
                }

                //fclose(fp);
                fp.close();
        } catch ( Exception e ) { System.out.println(e); return;}


/*      for (i=0;i<8;i++){
                System.out.println( sx_max[i] + " " + sy_max[i] + " " + lmax[i]);
        }
        for (i=0;i<ns;i++){
                System.out.print( dac[0][i] + " " + dac[1][i] + " " + dac[2][i]+ " ");
                System.out.println( sx[i] + " " + sy[i] + " " + l[i]);
        }
        System.out.println( sx_min + " " + sy_min + " " + lmin);
//*/

        //Make average for w, r,g,b.
        for (i=0;i<4;i++)
        {
                lmax[i] = (lmax[i]+lmax[i+4])/2;
                sx_max[i] = (sx_max[i]+sx_max[i+4])/2;
                sy_max[i] = (sy_max[i]+sy_max[i+4])/2;
        }

        xyzmin[1] = lmin;
        xyzmin[0] = xyzmin[1]/sy_min*sx_min;
        xyzmin[2] = xyzmin[1]/sy_min*(1.0-sy_min-sx_min);

                // 3 guns: r,g,b, xl[0][]-r scalar using matrix inversion.
        cm_plccMatrix_burns(sx_max[1],sy_max[1],sx_max[2],sy_max[2],
                                sx_max[3],sy_max[3],lmax,coefrgbxyz,coefxyzrgb);
        for (i=0;i<3;i++)
        {
                if(inter == 1)
                {
                        for(j=0;j<ns;j++)  //include interreflection.
                        {
                                xl[i][j] =((l[j]/sy[j]*sx[j])-xyzmin[0])*coefxyzrgb[i][0]
                                                +(l[j]-xyzmin[1])*coefxyzrgb
[i][1]
                                                +(l[j]/sy[j]*(1.0-sx[j]-sy[j])-
xyzmin[2])*coefxyzrgb[i][2];
                        }
                }
                else
                {
                        for(j=0;j<ns;j++)  //not include interreflection.
                        {
                                xl[i][j] =(l[j]/sy[j]*sx[j])*coefxyzrgb[i][0]
                                                +l[j]*coefxyzrgb[i][1]
                                                +l[j]/sy[j]*(1.0-sx[j]-sy[j])
*coefxyzrgb[i][2];
                        }
                }
        }
```

```
            //normalise xl so that white has xl = 1.0;
            for (i=0;i<3;i++)
                    for(j=0;j<ns;j++)
                            xl[i][j] /=xl[i][ns-1];
            calmonit_simplex(dac,ns,xl,gog); //gog[][0] - gamma, gog[][1]-gain

            valid = new Boolean(true);
            g_datavalid = new Boolean(true);
    }

    public static boolean IsdataValid(){
            return g_datavalid.booleanValue();
    }

    /* Burns' matrix */
    static void cm_plccMatrix_burns(double sxr,double syr,double sxg,double syg,double
sxb,
                    double syb,double lmax[],double coefrgbxyz[][],double coefxyzrgb[][])
    {
                    double a[][] = new double[3][3];
                    double  szr,szg,szb;
                    int i;

                    szr=1.0 - sxr - syr;
                    szg=1.0 - sxg - syg;
                    szb=1.0 - sxb - syb;
                    a[0][0] = sxr*lmax[1]/syr; a[1][0] = lmax[1]; a[2][0] = szr*lmax[1]/syr;
                    a[0][1] = sxg*lmax[2]/syg; a[1][1] = lmax[2]; a[2][1] = szg*lmax[2]/syg;
                    a[0][2] = sxb*lmax[3]/syb; a[1][2] = lmax[3]; a[2][2] = szb*lmax[3]/syb;

                    for ( i = 0; i < 3; i++ )
                    {
                                    coefrgbxyz[i][0] =a[i][0];
                                    coefrgbxyz[i][1] =a[i][1];
                                    coefrgbxyz[i][2] =a[i][2];
                    }
                    cm_plccInvMatrix(coefrgbxyz,coefxyzrgb);
    }

    /*
     * Calculating Inverse of 3x3 Matrix
     */
    public static void cm_plccInvMatrix(double a[][],double c[][])
    {
                    //double  b[3][3],sum;
                    double  b[][] = new double[3][3];
                    double sum;
                    int i,j;

                    b[0][0] = a[1][1]*a[2][2] - a[1][2]*a[2][1];
                    b[0][1] = a[1][2]*a[2][0] - a[1][0]*a[2][2];
                    b[0][2] = a[1][0]*a[2][1] - a[1][1]*a[2][0];
                    b[1][0] = a[0][2]*a[2][1] - a[0][1]*a[2][2];
                    b[1][1] = a[0][0]*a[2][2] - a[0][2]*a[2][0];
                    b[1][2] = a[0][1]*a[2][0] - a[0][0]*a[2][1];
                    b[2][0] = a[0][1]*a[1][2] - a[0][2]*a[1][1];
                    b[2][1] = a[0][2]*a[1][0] - a[0][0]*a[1][2];
                    b[2][2] = a[0][0]*a[1][1] - a[0][1]*a[1][0];
```

```
                sum = a[0][0]*(a[1][1]*a[2][2] - a[1][2]*a[2][1]);
                sum = sum - a[0][1]*(a[1][0]*a[2][2] - a[1][2]*a[2][0]);
                sum = sum + a[0][2]*(a[1][0]*a[2][1] - a[1][1]*a[2][0]);

                for ( i = 0; i < 3; i++ )
                for ( j = 0; j < 3; j++ )
                c[j][i] = b[i][j]/sum;
}

static void calmonit_simplex(int dac[][],int ns,double rgb[][],double co[][])
{

        //double *x,**p,*y,dd[18],scalar[18],ftol,f;
        double x[], p[][] ,y[];
        double dd[] = new double[18];
        double scalar[] = new double[18];
        double ftol,f;
        int i,j,m,np,mp,ndim;

        np=2;
        mp=np+1;
        ftol=0.000001;
        ndim=np;
        x = new double[np];
        y = new double[mp+1];
//      p = new (double(*[mp+1]));
//      for (i=0;i<=mp;i++)     p[i] = new double[np+1];
        p = new double[mp+1][np+1];

        for (m=0;m<3;m++)       {
                for (j=0;j<ns;j++)          {
                        dd[j]= dac[m][j]/255.;
                        scalar[j]=rgb[m][j];
                }
                for (i=1;i<=mp;i++)
                {
                        for (j=1;j<=np;j++)
                                x[j-1]=p[i][j]=(i == (j+1)? 2.0:0.8);
                        y[i] = calmonit_func(scalar,dd,ns,x);

                }
                calmonit_amoeba(p,y,ns,ndim,ftol,scalar,dd);
                for (i=1;i<=np;i++) {
                        co[m][i-1]=(p[1][i]+p[2][i]+p[3][i])/3.;
                }
        }

        //delete [] x;
        //delete [] y;
        //for (i=0;i<=mp;i++) delete [np+1] p[i];
        //delete [mp+1] p;
}

static double calmonit_func(double x[],double dac[],int ns,double y[])
{
        double s,xd, f;
        int i;
```

```
        s=0.0;
        for (i=0;i<ns;i++)          {
                if(y[1]*dac[i]+(1.0-y[1]) <= 0.0) xd = 0.0;
                else     xd= Math.pow(y[1]*dac[i]+1.0-y[1],y[0]);
                //printf("\nxd = %lf",xd);
                //s += (x[i] - pow(10.,xd))*(x[i] - pow(10.,xd));
                s += (x[i] - xd)*(x[i] -xd);
        }
        f =100.*Math.sqrt(s/ns);
        //fprintf(stdout, "\nf = %lf", *f);
        return f;
}

static void calmonit_amoeba(double p[][], double y[],int ns,int ndim, double ftol,
                        double xyz[],double dd[])

{
        int i, ihi,inhi,j,mpts=ndim+1,nfunk,ilo;
        double rtol,sum,swap,ysave,ytry,psum[];

        psum= new double[ndim];
        nfunk=0;
        //GET_PSUM;
        for (j=1;j<=ndim;j++) {
                for (sum=0.0,i=1;i<=mpts;i++) sum += p[i][j];
                psum[j-1] = sum;
        }

        for (;;) {
                ilo=1;
                //ihi = y[1]>y[2] ? (inhi=2,1) : (inhi=1,2);
                if (y[1]>y[2]){
                        inhi=2;
                        ihi =1;
                } else {
                        inhi=1;
                        ihi =2;
                }
                for (i=1;i<=mpts;i++) {
                        if (y[i] <= y[ilo]) ilo=i;
                        if (y[i] > y[ihi]) {
                                inhi=ihi;
                                ihi=i;
                        } else if (y[i] > y[inhi] && i != ihi) inhi=i;
                }
                rtol=2.0*Math.abs(y[ihi]-y[ilo])/(Math.abs(y[ihi]+y[ilo]));
                if (rtol < ftol) {
                        //SWAP(y[1],y[ilo])
                        swap = y[1];
                        y[1] = y[ilo];
                        y[ilo] = swap;
                        //for (i=1;i<=ndim;i++) SWAP(p[1][i],p[ilo][i])
                        for (i=1;i<=ndim;i++){
                                swap = p[1][i];
                                p[1][i] = p[ilo][i];
                                p[ilo][i] = swap;
                        }
```

```
                        //printf("\niteration = %d\n",nfunk);
                        break;
                    }
                    //if (nfunk >= NMAX) fprintf(stderr,"NMAX exceeded");
                    nfunk +=2;
                    ytry=calmonit_amotry(p,y,psum,ns,ndim,xyz,dd,ihi,-1.0);
                    if (ytry <=y[ilo])
                        ytry=calmonit_amotry(p,y,psum,ns,ndim,xyz,dd,ihi,2.0);
                    else if (ytry >= y[inhi]) {
                        ysave=y[ihi];
                        ytry=calmonit_amotry(p,y,psum,ns,ndim,xyz,dd,ihi,0.5);
                        if (ytry >= ysave) {
                            for (i=1;i<=mpts;i++) {
                                if (i != ilo) {
                                    for (j=1;j<=ndim;j++)
                                        p[i][j]=psum[j-1]=0.5*(p[i][j]
+p[ilo][j]);
                                    y[i]=calmonit_func(xyz,dd,ns,psum);
                                }
                            }
                            nfunk += ndim;
                            //GET_PSUM
                            for (j=1;j<=ndim;j++) {
                                for (sum=0.0,i=1;i<=mpts;i++) sum += p[i][j];
                                psum[j-1] = sum;
                            }
                        }
                    } else --(nfunk);
                }
                //psum=float(psum);
                //delete [] psum;
            }

            static double calmonit_amotry(double p[][], double y[], double psum[],int ns, int ndim,
                double xyz[],double dd[],int ihi, double fac)
            {
                int j;
                double fac1,fac2,ytry,ptry[];

                ptry = new double[ndim];
                fac1=(1.0-fac)/ndim;
                fac2=fac1-fac;
                for (j=1;j<=ndim;j++) ptry[j-1]=psum[j-1]*fac1-p[ihi][j]*fac2;
                ytry = calmonit_func(xyz,dd,ns,ptry);
                if (ytry < y[ihi]) {
                    y[ihi]=ytry;
                    for (j=1;j<=ndim;j++) {
                        psum[j-1] += ptry[j-1]-p[ihi][j];
                        p[ihi][j]=ptry[j-1];
                    }
                }
                //delete [] ptry;
                return ytry;
            }

            public static void main(String[] args)
            {
                calibrate();
```

```
                for( int i=0; i<3; i++ )
                        for( int j=0; j<3; j++ )
                                System.out.println( g_coefrgb2xyz[i][j]);


/*              double m_coefxyz2rgb[][] = new double[3][3];
                System.out.println("Hello World!"+ g_coefxyz2rgb.length);
                for( int i=0; i< g_coefxyz2rgb.length; i++){
                        g_coefxyz2rgb[i][0]=i*3+1;
                        g_coefxyz2rgb[i][1]=i*3+2;
                        g_coefxyz2rgb[i][2]=i*3+3;
                        //m_coefxyz2rgb[i][i]=i;

                }
                System.arraycopy(g_coefxyz2rgb,0,m_coefxyz2rgb,0,m_coefxyz2rgb.length);
                for(int i=0; i< g_coefxyz2rgb.length; i++){
                        for(int j=0; j< g_coefxyz2rgb.length; j++){
                                System.out.print( g_coefxyz2rgb[i][j]+ " " );
                        }
                        System.out.println();
                }
                for(int i=0; i< g_coefxyz2rgb.length; i++){
                        for(int j=0; j< g_coefxyz2rgb.length; j++){
                                System.out.print( m_coefxyz2rgb[i][j]+ " " );
                        }
                        System.out.println();
                }
//*/
        }
}
```

```
/*      //global value
        extern bool   g_datavalid;
        extern double g_coefxyz2rgb[3][3], g_coefrgb2xyz[3][3];
        extern double g_gog[3][2], g_sx_max[8], g_sy_max[8];
        extern double g_xmin, g_ymin, g_zmin;
// */

class CColorSpace
{
        double m_X0, m_Y0, m_Z0;  //bright light
        int m_rgb[]; //space r,g,b => m_rgb[0],m_rgb[1],m_rgb[2]
        double m_XYZ[]; //space X,Y,Z => m_XYZ[0],m_XYZ[1],m_XYZ[2]
        //double m_h, m_L, m_C; //space h, L*, C*
        //double m_a, m_b;      //space L, a*, b*
        Double m_h, m_L, m_C; //space h, L*, C*
        Double m_a, m_b;      //space L, a*, b*
        double m_gog[][]; //[3][2] GOG modal
        double m_coefrgb2xyz[][];  //[3][3]coeffication from space RGB to XYZ
        double m_coefxyz2rgb[][];  //[3][3]coeffication from space XYZ to RGB
   double sy_max[], sx_max[]; //[8]
        double m_xmin, m_ymin, m_zmin;
        boolean m_boutrange;

    //construction
        public CColorSpace(){
                m_rgb = new int[3]; //space r,g,b => m_rgb[0],m_rgb[1],m_rgb[2]
                m_XYZ = new double[3]; //space X,Y,Z => m_XYZ[0],m_XYZ[1],m_XYZ[2]
                m_gog= new double[3][2]; //[3][2] GOG modal
                m_coefrgb2xyz = new double[3][3];  //[3][3]coeffication from space RGB to
XYZ
                m_coefxyz2rgb = new double[3][3];  //[3][3]coeffication from space XYZ to
RGB
                sy_max = new double[8];
                sx_max = new double[8]; //[8]
        }


        //Set and Get method in color space r,g,b
        public void Set_rgb(int rgb[]) {m_rgb[0] = rgb[0];m_rgb[1] = rgb[1];m_rgb[2] = rgb[2];}
        public void Get_rgb(int rgb[]) {rgb[0] = m_rgb[0];rgb[1] = m_rgb[1];rgb[2] = m_rgb[2];}

        //Set and Get method in color space XYZ
        public void Set_XYZ(double XYZ[]) {m_XYZ[0] = XYZ[0];m_XYZ[1] = XYZ[1];m_XYZ
[2] = XYZ[2];}
        public void Get_XYZ(double XYZ[]) {XYZ[0] = m_XYZ[0];XYZ[1] = m_XYZ[1];XYZ[2] =
m_XYZ[2];}

        //Set and Get method in color space hLC
        public void Set_h(double h) {m_h = new Double(h);}
        public void Set_L(double L) {m_L = new Double(L);}
        public void Set_C(double C) {m_C = new Double(C);}

        public double Get_h() {return m_h.doubleValue();}
        public double Get_L() {return m_L.doubleValue();}
        public double Get_C() {return m_C.doubleValue();}

        //Set and Get method in color space Lab
```

```
        public void Set_a(double a) {m_a = new Double(a);}
        public void Set_b(double b) {m_b = new Double(b);}

        public double Get_a()  {return m_a.doubleValue();}
        public double Get_b()  {return m_b.doubleValue();}

        //space convertion method
/*      public boolean ColorXYZ2rgb();
        public boolean Colorrgb2XYZ();

        public boolean ColorXYZ2LabCh();
        public boolean ColorLab2XYZ();
        public boolean ColorLch2XYZ();
//*/
        public boolean IsOutOfRange() {return m_boutrange;};


        public void Inilize()
        {

                m_boutrange = false;

                m_xmin = 0.0; //0.256;
                m_ymin = 0.0; //0.262;
                m_zmin = 0.0;  //0.2215;
                //sRGB coeffication of the rgb=>xyz convertion
                m_coefrgb2xyz[0][0] = 41.23809;
                m_coefrgb2xyz[0][1] = 35.75728;
                m_coefrgb2xyz[0][2] = 18.04523;
                m_coefrgb2xyz[1][0] = 21.26199;
                m_coefrgb2xyz[1][1] = 71.51388;
                m_coefrgb2xyz[1][2] = 7.21499;
                m_coefrgb2xyz[2][0] = 1.93435;
                m_coefrgb2xyz[2][1] = 11.92122;
                m_coefrgb2xyz[2][2] = 94.05066;

                //calculate the inverse matrix of m_coefrgb2xyz
                GlobalVar.cm_plccInvMatrix(m_coefrgb2xyz,m_coefxyz2rgb);

                //calculate the bright light coeff
                sy_max[0] = 0.3290;
                sx_max[0] = 0.3127;

                m_X0 = 100.0/sy_max[0]*sx_max[0];
                m_Y0 = 100.0;
                m_Z0 = 100.0/sy_max[0]*(1.0-sx_max[0]-sy_max[0]);

                //GOG modal, gog[][0]->gamma, gog[][1]->gain, 1-gog[][1]->offset
                m_gog[0][0] = 2.4;
                m_gog[0][1] = 1.055;
                m_gog[1][0] = 2.4;
                m_gog[1][1] = 1.055;
                m_gog[2][0] = 2.4;
                m_gog[2][1] = 1.055;

                m_X0 = 100.0/sy_max[0]*sx_max[0];
                m_Y0 = 100.0;
                m_Z0 = 100.0/sy_max[0]*(1.0-sx_max[0]-sy_max[0]);
```

```
                //read the data calibrated from the monitor.txt file
                if( GlobalVar.IsdataValid())
                {

                        //memcpy(m_coefxyz2rgb, g_coefxyz2rgb, sizeof(m_coefxyz2rgb));
                        //memcpy(m_coefrgb2xyz, g_coefrgb2xyz, sizeof(m_coefrgb2xyz));
                        //memcpy(m_gog, g_gog, sizeof(m_gog));
                        //memcpy(sx_max, g_sx_max, sizeof(sx_max));
                        //memcpy(sy_max, g_sy_max, sizeof(sy_max));

                        System.arraycopy
(GlobalVar.g_coefxyz2rgb,0,m_coefxyz2rgb,0,m_coefxyz2rgb.length);
                        System.arraycopy
(GlobalVar.g_coefrgb2xyz,0,m_coefrgb2xyz,0,m_coefrgb2xyz.length);
                        System.arraycopy(GlobalVar.g_gog,0,m_gog,0,m_gog.length);
                        System.arraycopy(GlobalVar.g_sx_max,0, sx_max,0,
sx_max.length);
                        System.arraycopy(GlobalVar.g_sy_max,0, sy_max,0,
sy_max.length);

                        m_xmin = GlobalVar.g_xmin;
                        m_ymin = GlobalVar.g_ymin;
                        m_zmin = GlobalVar.g_zmin;
                }
        }



        //space match method
        public boolean ColorXYZ2rgb()
        {

                Integer inGamt = new Integer(0);
//              cm_plccXYZ2RGB(m_XYZ[0],m_XYZ[1],m_XYZ
[2],m_coefxyz2rgb,m_gog,m_xmin, m_ymin, m_zmin,
//                                                      &m_rgb[0],&m_rgb[1],&m_rgb[2],
&inGamt);
                m_boutrange = cm_plccXYZ2RGB(m_XYZ[0],m_XYZ[1],m_XYZ
[2],m_coefxyz2rgb,m_gog,m_xmin, m_ymin, m_zmin,
                                                        m_rgb, inGamt);

                //m_boutrange = (inGamt.intValue() == 0)?true:false;
                //System.out.println(m_boutrange);
                return true;
        }

        /*
         * Convert XYZ to Monitor RGB
         */
        boolean cm_plccXYZ2RGB(double X,double Y,double Z,double coefxyzrgb[][],double
gog[][],
                                                        double xmin,double ymin,double
zmin,int RGB[], Integer inGamt)
        {
                        //double  rgb[3],xyzc[3],dacrgb[3];
                        double  rgb[] = new double[3];
                        double  xyzc[] = new double[3];
```

```
        double  dacrgb[] = new double[3];

        int i,j;

        xyzc[0] = X -xmin;
        xyzc[1] = Y -ymin;
        xyzc[2] = Z -zmin;
        inGamt = new Integer(1);
        for ( i = 0; i < 3; i++ )
        {
                rgb[i] = 0.0;
                for ( j = 0; j < 3; j ++)
                        rgb[i] = rgb[i] + coefxyzrgb[i][j]*xyzc[j];

                if ( rgb[i] <= 0.0 )
                {
                        rgb[i] = 0.0;
                        inGamt = new Integer(0);
                }
                if ( rgb[i] >= 1.0 )
                {
                        rgb[i] = 1.0;
                        inGamt = new Integer(0);
                }
        }

        // GOG method
        if (GlobalVar.IsdataValid()) // if can read file
        {
                for ( i = 0; i < 3 ; i++ )
                {
                        if (rgb[i] == 0.0 ) dacrgb[i] = 0.0;
                        else
                        dacrgb[i] = 255.0*(Math.pow(rgb[i],1.0/gog[i][0]) -
                                                        (1.0-gog[i][1]))/gog[i]
[1];
                        if (dacrgb[i] < 0.0) dacrgb[i] = 0.0;
                }
        } else { // can't read file [use sRGB system]
                for ( i = 0; i < 3 ; i++)
                {
                        if (rgb[i] <= 0.00304 ) dacrgb[i] = 255.0*12.92*rgb[i];
                        else
                        dacrgb[i] = 255.0*(1.055*Math.pow(rgb[i],1.0/2.4)-
0.055);
                        if (dacrgb[i] < 0.0) dacrgb[i] = 0.0;
                }
        }


        //RGB[0] = (int) floor(dacrgb[0] + 0.5);
        RGB[0] = (int) (dacrgb[0] + 0.5);
        RGB[1] = (int) (dacrgb[1] + 0.5);
        RGB[2] = (int) (dacrgb[2] + 0.5);

        return ( inGamt.intValue() == 0 );
    }
```

```
public boolean Colorrgb2XYZ()
{

        //int inGamt = 0;
        Integer inGamt = new Integer(0);
        cm_plccRGB2XYZ(m_rgb[0],m_rgb[1],m_rgb[2],m_coefrgb2xyz,m_gog,
                        //&m_XYZ[0],&m_XYZ[1],&m_XYZ[2],&inGamt);
                        m_XYZ, inGamt);
        return true;
}

/*
 * Convert Monitor RGB  to XYZ
 */
void cm_plccRGB2XYZ(int R,int G,int B,double coefrgbxyz[][],double gog[][],
                        double XYZ[], Integer inGamt)
{
                //double  rgb[3],xyzc[3],drgb[3],temp;
                double  rgb[] = new double[3];
                double  xyzc[] = new double[3];
                double  drgb[] = new double[3];
                double temp;
                int i,j;


                drgb[0] =  (double) R;
                drgb[1] =  (double) G;
                drgb[2] =  (double) B;
                inGamt = new Integer(1);

                if (GlobalVar.IsdataValid())  // if can read file
                {
                        for (  i = 0; i < 3; i++ )
                        {
                                // GoG method
                                temp = gog[i][1]*drgb[i]/255.0 + 1.0-gog[i][1];
                                if (temp >= 0.0)
                                        rgb[i] = Math.pow(temp,gog[i][0]);
                                else
                                        rgb[i] = 0.0;
                        }
                } else { // can't read file [use sRGB system]
                        for (  i = 0; i < 3; i++ )
                        {
                                temp = drgb[i]/255.0;
                                if ( temp <= 0.03928 ){ rgb[i] = temp/12.92;}
                                if ( temp > 0.03928){ rgb[i] = Math.pow((temp+
0.055)/1.055,2.4);}
                                if ( temp < 0.0){ rgb[i] = 0.0;}
                        }
                }

                for ( i = 0; i < 3; i++)
                {
                        xyzc[i] = 0.0;
                        for ( j = 0; j < 3; j++ )
                                xyzc[i] = xyzc[i] + coefrgbxyz[i][j]*rgb[j];
```

```
                }
                XYZ[0] = xyzc[0];
                XYZ[1] = xyzc[1];
                XYZ[2] = xyzc[2];
        }       //void cm_plccRGB2XYZ

        public boolean ColorXYZ2LabCh()
        {
                //cm_CIELabch(m_X0,m_Y0,m_Z0,m_XYZ[0],m_XYZ[1],m_XYZ[2],&m_L,
&m_a,&m_b,&m_C,&m_h);
                cm_CIELabch(m_X0,m_Y0,m_Z0,m_XYZ[0],m_XYZ[1],m_XYZ
[2],m_L,m_a,m_b,m_C,m_h);
                return true;
        }

        /*
         *
         *      1976 CIE L*a*b* Uniform Colour Space
         *          L*, a*, b*, C*, h coordinates
         *      Input parameters : X, Y, Z
         *              X0 Y0 Z0 (illuminant)
         *      Output parameters : L(L*), a(a*), b(b*), c(c*) and h
         */
        void cm_CIELabch(double X0,double Y0,double Z0,double X,double Y,double Z,
//                                      double *L,double *a,double *b,double *c,double *h)
                                        Double L,Double a,Double b,Double c,Double h)

        {
                //double ILL[3],XYZ[3],FN[3];
                double ILL[] = new double[3];
                double XYZ[] = new double[3];
                double FN[] = new double[3];
                int i;

                ILL[0] = X0;
                ILL[1] = Y0;
                ILL[2] = Z0;
                XYZ[0] = X;
                XYZ[1] = Y;
                XYZ[2] = Z;
                for ( i = 0; i < 3; i++)
                if ( XYZ[i]/ILL[i]  < 0.008856 )
                        FN[i] = 7.787*XYZ[i]/ILL[i] + 16.0/116.0;
                else
                        FN[i] = Math.pow(XYZ[i]/ILL[i], 1.0/3.0);
                L = new Double(116.0*FN[1] - 16.0);
                //a = new Double(500.0*(FN[0] - FN[1]));
                m_a = new Double(500.0*(FN[0] - FN[1]));
                //b = new Double(200.0*(FN[1] - FN[2]));
                m_b = new Double(200.0*(FN[1] - FN[2]));
                c = new Double(Math.sqrt(Math.pow(m_a.doubleValue(),2.0) +
Math.pow(m_b.doubleValue(),2.0)));
                h = new Double(cm_arctan(m_a.doubleValue(),m_b.doubleValue()));
                m_C = new Double(Math.sqrt(Math.pow(m_a.doubleValue(),2.0) +
Math.pow(m_b.doubleValue(),2.0)));
                m_h = new Double(cm_arctan(m_a.doubleValue(),m_b.doubleValue
()));
        } // void cm_CIELabch
```

```
/*
 *      Calculating the arctangent function
 *      h = arctan ( b/a )
 */
double cm_arctan(double a,double b)
{
      double  h;

      if ( a == 0.0 )
      {
            if ( b > 0.0)
                  return(90.0);
            else if ( b < 0 )
                  return(270.0);
      }
      if ( b == 0.0 )
      {
            if ( a >= 0.0 )
                  return(0.0);
            else if ( a < 0 )
                  return(180.0);
      }
      h = Math.atan(b/a)*180.0/Math.PI;
      if ( a < 0.0 )
            h = h + 180.0;
      else if ( b < 0.0 )
           h = h + 360.0;
      return(h);
}

          public boolean ColorLab2XYZ()
          {
                    cm_CIELab2XYZ(m_X0,m_Y0,m_Z0,m_L.doubleValue(),m_a.doubleValue
(),m_b.doubleValue(),m_XYZ);
                    return true;
          }

          /*
           *     Transforming the 1976 CIE L*a*b*, L*, a* and b*
           *          values to the tristimulus values
           *     Input parameters : L (L*),  a (a*), b (b*)
           *                        X0 Y0 Z0 (illuminant)
           *     Output parameters : X, Y, Z
           */
          void cm_CIELab2XYZ(double X0,double Y0,double Z0,double L,double a,double b,
                                                    double XYZarg[])
          {
                    //double  FN[3], ILL[3], XYZ[3];
                    double  FN[] = new double[3];
                    double  ILL[] = new double[3];
                    double  XYZ[] = new double[3];

                    int i;

                    ILL[0] = X0;
                    ILL[1] = Y0;
                    ILL[2] = Z0;
                    FN[1]  = L/116.0;
```

```
                FN[0]  = FN[1] + a/500.0;
                FN[2]  = FN[1] - b/200.0;
                for (i = 0; i < 3; i++)
                        if ( FN[i] <= 0.068961672 )
                                XYZ[i] = ILL[i]*FN[i]/7.787;
                        else   XYZ[i] = ILL[i]*Math.pow(FN[i] + 16.0/116.0, 3.0);
                XYZarg[0] = XYZ[0];
                XYZarg[1] = XYZ[1];
                XYZarg[2] = XYZ[2];
        }


        public boolean ColorLch2XYZ()
        {
                cm_CIELch2XYZ(m_X0,m_Y0,m_Z0,m_L.doubleValue(),m_C.doubleValue
(),m_h.doubleValue(),m_XYZ);
                return true;
        }

        /*
         *    Transforming the 1976 CIE L*a*b*, L*, c* and h
         *         values to the tristimulus values
         *    Input parameters : L (L*), c (c*), h
         *                X0 Y0 Z0 (illuminant)
         *    Output parameters : X, Y, Z
         */
        void cm_CIELch2XYZ(double X0,double Y0,double Z0,double L,double c,double h,
                                                double XYZ[])
        {
                double  a,b,d;

                d = h*Math.PI/180;
                a  = c*Math.cos(d);
                b  = c*Math.sin(d);
                cm_CIELab2XYZ(X0,Y0,Z0,L,a,b,XYZ);
        }


};
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//import java.applet.*;
import javax.swing.event.*;
import java.util.*;
import java.io.*;

class CColorConvertDlg  extends JFrame implements ActionListener, WindowListener,
MouseListener, MouseMotionListener, ChangeListener
{

        private static final int FRAME_WIDTH = 700;
        private static final int FRAME_HEIGHT = 575;

        private static final int FRAME_X_ORIGIN = 50;
        private static final int FRAME_Y_ORIGIN = 0;

        private static final int BUTTON_WIDTH = 150;
        private static final int BUTTON_HEIGHT = 30;

        static final int LENGTH_LHC_C = 100;
        static final int LENGTH_LHC   = 100;
        static final int LENGTH_LAB   = 100;

        static final int ROW_SAMPLE   = 4;
        static final int COL_SAMPLE   = 4;

        static final int ratio_c      = 2;//(200.0 / LENGTH_LHC_C)
        static final Color bgcolor = new Color( 0xCC, 0xCC, 0xCC );

        static int m_Hold = 0;
        static int m_Lold = LENGTH_LHC-60;

        public static void SaveColorEmotion(double L, double h, double C, PrintWriter pout){
                CalculateColorEmotion( L, h, C, false);
                //(float)m_warm, (float)m_light, (float)m_deep, (float)m_heavy, (float)m_vivid,
                //(float)m_gaudy, (float)m_striking, (float)m_dynamic, (float)m_distinct, (float)
m_transparent, (float)m_soft,(float)m_strong);
                pout.print((float)m_warm+" "+ (float)m_light+" "+ (float)m_deep+" "+ (float)
m_heavy+" "+ (float)m_vivid);
                pout.println((float)m_gaudy+" "+ (float)m_striking+" "+ (float)m_dynamic+" "+
(float)m_distinct+" "+ (float)m_transparent+" "+ (float)m_soft+" "+(float)m_strong);
        }

        public static void CalculateColorEmotion(double L, double h, double C, boolean
bRefresh){

                double  deep;
                double  distinct;
                double  dynamic;
                double  gaudy;
                double  heavy;
                double  light;
                double  soft;
                double  striking;
                double  strong;
                double  transparent;
```

```
                double  vivid;
                double  warm;
/*
                if(h <=180 && h >= 0)
                        warm = 0.154*L + 39.378*Math.pow(C,0.372) - 0.303*h - 113.855;
                else
                        warm = 0.355 *L + 23.476*Math.pow(C,0.429) - 0.159*(360 - h) -
105.710;

                if(h <=180 && h >= 0)
                        light = 2.121*L + 1.798*C - 0.022*h - 175.642;
                else
                        light = 2.295*L + 2.886*C + 0.117*(360 - h) - 198.033;

                if(h <=180 && h >= 0)
                        deep = -3.590*L + 0.451*C + 0.040*h + 189.467;
                else
                        deep = -3.674*L - 0.216*C + 0.098*(360 - h) + 189.127;

                if(h <=180 && h >= 0)
                        heavy = -3.340*L + 0.476*C + 0.037*h + 175.467;
                else
                        heavy = -3.477*L - 0.264*C + 0.072*(360 - h) + 182.866;

                if(h <=180 && h >= 0)
                        vivid = 0.713*L + 11.209*Math.pow(C,0.668) + 0.009*h - 147.825;
                else
                        vivid = 0.982*L + 14.643*Math.pow(C,0.660) + 0.091*(360 - h) -
165.574;

                if(h <=180 && h >= 0)
                        gaudy = -0.332*L + 4.574*Math.pow(C,0.867) - 0.081*h - 73.973;
                else
                        gaudy = -0.114*L + 13.195*Math.pow(C,0.664) - 0.005*(360 - h) -
103.481;

                if(h <=180 && h >= 0)
                        striking = -0.750*L + 2.847*Math.pow(C,0.961) - 0.052*h - 35.564;
                else
                        striking = -0.748*L + 10.597*Math.pow(C,0.684) + 0.060*(360 - h) -
54.801;

                if(h <=180 && h >= 0)
                        dynamic = -0.296*L + 3.162*Math.pow(C,0.931) - 0.073*h - 68.835;
                else
                        dynamic = -0.120*L + 4.385*Math.pow(C,0.864) + 0.032*(360 - h) -
84.791;

                if(h <=180 && h >= 0)
                        distinct = 1.820*L + 1.214*C + 0.077*h - 144.740;
                else
                        distinct = 1.746*L + 2.393*C + 0.111*(360 - h) - 147.418;

                if(h <=180 && h >= 0)
                        transparent = 2.363*L + 0.876*C + 0.040*h - 180.640;
                else
                        transparent = 2.119*L + 1.863*C + 0.072*(360 - h) - 169.497;
```

```
                if(h <=180 && h >= 0)
                        soft = 2.900*L - 0.510*C - 0.051*h - 146.700;
                else
                        soft = 2.953*L + 0.424*C - 0.020*(360 - h) - 159.795;

                if(h <=180 && h >= 0)
                        strong = -2.625*L + 1.185*C + 0.053*h + 116.320;
                else
                        strong = -2.758*L + 0.353*C + 0.050*(360 - h) + 135.877;
*/
                double dh40 = 40-h;
                double dh290 = h+70;
//              if ( (0<h) && (h<=110) )   dh290 = h+70;
                if ( (110<h) && (h<=290) ) dh290 = 290 - h;
                if ( (290<h) && (h<=360) ) dh290 = h - 290;

//              if ( (0<h) && (h<=40) )   dh40 = 40 - h;
                if ( (40<h) && (h<=220) ) dh40 = h - 40;
                if ( (220<h) && (h<=360) ) dh40 = 400 - h;

                double one_dhC = (1 - dh290/360) * C;

                light           = Math.pow( Math.pow(3.4*(L-10),2) + Math.pow(4.5
*one_dhC,2), 0.5 ) - 184;
                soft            = - Math.pow( Math.pow(2.2*(L-90),2) + Math.pow(0.9
*one_dhC,2),0.5) + 79;
                warm            = Math.pow( Math.pow(0.27*(L-100),2) + Math.pow(1.48*(1
+Math.cos(Math.toRadians(dh40)))*one_dhC,2),0.5) - 58;
                transparent     = Math.pow( Math.pow(3.1*(L-30),2) + Math.pow(2.7
*one_dhC,2),0.5) - 122;
                deep            = Math.pow( Math.pow(2.6*(L-100),2) + Math.pow(1.8
*one_dhC,2),0.5) - 90;
                distinct = Math.pow( Math.pow(1.9*(L-60),2) + Math.pow(3.3*one_dhC,2),0.5)
- 62;
                heavy           = Math.pow( Math.pow(2.6*(L-100),2) + Math.pow(0.6
*one_dhC,2),0.5) - 96;
                vivid           = Math.pow( Math.pow(2.2*(L-10),2) + Math.pow(5
*one_dhC,2),0.5) - 157;
                strong          = Math.pow( Math.pow(2.1*(L-90),2) + Math.pow(0.6
*one_dhC,2),0.5) - 52;
                dynamic         = Math.pow( Math.pow(1.1*(L-20),2) + Math.pow(3.8
*one_dhC,2),0.5) - 100;
                gaudy           = Math.pow( Math.pow(0.4*(L-10),2) + Math.pow(3.8
*one_dhC,2),0.5) - 95;
                striking = Math.pow( Math.pow(1.6*(L-90),2) + Math.pow(3.1*one_dhC,2),0.5)
- 65;

                //double emotion[12] = {0};
                double emotion[] = new double[12];
                emotion[0] = deep;
                emotion[1] = distinct;
                emotion[2] = dynamic;
                emotion[3] = gaudy;
                emotion[4] = heavy;
                emotion[5] = light;
                emotion[6] = soft;
                emotion[7] = striking;
```

```
                emotion[8] = strong;
                emotion[9] = transparent;
                emotion[10] = vivid;
                emotion[11] = warm;

                for(int i = 0; i < 12; i++)
                {
                        if(emotion[i] < -100)
                                emotion[i] = -100.0;
                        else if(emotion[i] > 100.0)
                                emotion[i] = 100.0;
                        emotion[i] = (int)(emotion[i] * 100) / 100.0;

                }

                m_deep    = emotion[0];
                m_distinct        = emotion[1];
                m_dynamic       = emotion[2];
                m_gaudy           = emotion[3];
                m_heavy           = emotion[4];
                m_light     = emotion[5];
                m_soft      = emotion[6];
                m_striking        = emotion[7];
                m_strong          = emotion[8];
                m_transparent =           emotion[9];
                m_vivid     = emotion[10];
                m_warm            = emotion[11];

                if(bRefresh)
                        UpdateData(false);


        }

        public void stateChanged(ChangeEvent e){
                if (e.getSource() == m_sldH)
                {
                                if ( m_Hold > m_sldH.getValue())
                                {
                                        m_Hold = m_sldH.getValue();
                                        m_Hold = m_Hold - (m_Hold % 10);
                                        m_sldH.setValue(m_Hold);
                                } else if( m_Hold < m_sldH.getValue())
                                {
                                        m_Hold = m_sldH.getValue();
                                        m_Hold = m_Hold -( m_Hold % 10) + ( ((m_Hold %
10)==0)?0:10 );
                                        m_sldH.setValue(m_Hold);
                                }
                                System.out.println( "change" );
                 } else if(e.getSource() == m_sldL){
                                if ( m_Lold > m_sldL.getValue())
                                {
                                        m_Lold = m_sldL.getValue();
                                        m_Lold = m_Lold - (m_Lold % 10);
                                        m_sldL.setValue(m_Lold);
                                } else if( m_Lold < m_sldL.getValue())
```

```
                                {
                                        m_Lold = m_sldL.getValue();
                                        m_Lold = m_Lold -( m_Lold % 10) + ( ((m_Lold % 10)
==0)?0:10 );
                                        m_sldL.setValue(m_Lold);
                                }
                                System.out.println( "change" );
                        }
                }

                void onClickedRectLab(Point point){
                        //setTitle("Lab");
                        int rgb[]= new int[3];
                        int RGB[]= new int[3];
                        int i,j,k;
                        int cx, cy;
                        cx = LENGTH_LAB;
                        cy = LENGTH_LAB;

                        i = (point.x - cx);
                        j = (point.y - cy);
                        j =  - j;

                        m_colorLab[LENGTH_LAB + i][LENGTH_LAB+j].Get_rgb(rgb);
                        for(k = 0; k < 3; k++)
                        {
                                RGB[k] = (int)(rgb[k]);
                        }

                        if(!m_colorLab[LENGTH_LAB + i][LENGTH_LAB+j].IsOutOfRange() &&
                                max(RGB[0], max(RGB[1], RGB[2])) <= 255 && min(RGB[0], min
(RGB[1], RGB[2])) >= 0)
                        {
                                m_ptLabOld = m_ptLab;
                                m_ptLab = point;
//                              rectOld = CRect(m_ptLabOld.x - 4, m_ptLabOld.y - 4, m_ptLabOld.x
+ 5, m_ptLabOld.y + 5);
//                              rect    = CRect(m_ptLab.x - 4, m_ptLab.y - 4, m_ptLab.x + 5,
m_ptLab.y + 5);
                                m_clrMarkLab = new Color(255-RGB[0], 255-RGB[1],255-RGB[2]);
                                m_bHasClicked = true;
                                m_rectLab.setm_bHasClicked(true);
                                m_rectLhc.setm_bHasClicked(true);
                                m_rectLab.setPointMark(point);
                                m_rectLab.setClrMark( m_clrMarkLab );
                                m_rectLab.repaint();



//                              InvalidateRect(&rectOld);
//                              InvalidateRect(&rect);

//                              CClientDC dc(this);

                                m_clrSelected = new Color(RGB[0], RGB[1], RGB[2]);
                                m_lhcSelected[0] = m_colorLab[LENGTH_LAB + i]
[LENGTH_LAB+j].Get_L();
                                m_lhcSelected[1] = m_colorLab[LENGTH_LAB + i]
```

```
[LENGTH_LAB+j].Get_h();
                        m_lhcSelected[2] = m_colorLab[LENGTH_LAB + i]
[LENGTH_LAB+j].Get_C();

                        //InvalidateRect(&m_rectSample);
                        m_rectSample.setBackground( m_clrSelected );

                        double XYZ[] = new double[3];
                        double xy[] = new double[2];
                        m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_XYZ(XYZ);
                        for(k = 0; k < 3; k++)
                        {
                                XYZ[k] = ((int)(XYZ[k] * 10000)) / 10000.0;
                        }
                        cm_XYZ2xy(XYZ[0],XYZ[1],XYZ[2],xy);

                        xy[0] = ((int)(xy[0] * 10000)) / 10000.0;
                        xy[1] = ((int)(xy[1] * 10000)) / 10000.0;

                        m_X = XYZ[0];
                        m_Y = XYZ[1];
                        m_Z = XYZ[2];

                        m_lx = xy[0];
                        m_ly = xy[1];

                        m_red = (int)(rgb[0]);
                        m_green = (int)(rgb[1]);
                        m_blue = (int)(rgb[2]);

                        m_l = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_L();
                        m_h = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_h();
                        m_c = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_C();
                        m_a = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_a();
                        m_b = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_b();
                        m_a = ((int)(m_a * 10000)) / 10000.0;
                        m_b = ((int)(m_b * 10000)) / 10000.0;


                        CalculateColorEmotion(m_l, m_h, m_c,true);

                        m_sldA.setValue((int)m_a);
                        m_sldB.setValue(-(int)m_b);


                        //reflesh the space of the Lhc
                        //for draw the mark point of Lab
                        m_sldH.setValue((int)(m_h+0.5));
                        m_sldC.setValue((int)(m_c+0.5));
                        m_sldL.setValue(LENGTH_LHC - (int)(m_l+0.5));

                        m_c = ((int)(m_c * 10000)) / 10000.0;
                        m_h = ((int)(m_h * 10000)) / 10000.0;

                        UpdateData(false);
```

```
                        m_ptLhc = new Point((int)(m_c*ratio_c+0.5), (LENGTH_LHC - (int)
(m_l+0.5))*2);
                        m_rectLhc.setPointMark(m_ptLhc);
                        m_clrMarkLhc = new Color(255-RGB[0], 255-RGB[1],255-RGB[2]);
                        m_rectLhc.setClrMark( m_clrMarkLhc );
                        //BeginWaitCursor();
                        DrawLhcGraph(m_h);
                        //dc.BitBlt(m_rectLhc.left, m_rectLhc.top, m_rectLhc.Width(),
m_rectLhc.Height(), &m_LhcDC, 0, 0, SRCCOPY);
                        //EndWaitCursor();



                        //InvalidateRect(&m_rectLhc);
//*/
                }
        }


        void onClickedRectLhc(Point point){
                int rgb[]= new int[3];
                int RGB[]= new int[3];
                int i,j,k;
                //setTitle("x="+point.getX()+" y="+point.getY());

                //calculate the address, caution is the address of j is in opposite
                i = (int)(point.getX() / (ratio_c) + 0.5);
                j = (int)(point.getY() / 2);
                j = LENGTH_LHC - j;  //reverse the y axis

                //get and display rgb
                m_colorLhc[i][j].Get_rgb(rgb);
                for(k = 0; k < 3; k++)
                {
                        RGB[k] = (int)(rgb[k]);
                }

                if(!m_colorLhc[i][j].IsOutOfRange() && max(RGB[0], max(RGB[1], RGB[2]))
<= 255 && min(RGB[0], min(RGB[1], RGB[2])) >= 0)
                {
                        //color and invalide area of mark
//                      m_ptLhcOld = m_ptLhc;
//                      m_ptLhc = point;
//                      rectOld = CRect(m_ptLhcOld.x - 4, m_ptLhcOld.y - 4, m_ptLhcOld.x
+ 5, m_ptLhcOld.y + 5);
//                      rect    = CRect(m_ptLhc.x - 4, m_ptLhc.y - 4, m_ptLhc.x + 5,
m_ptLhc.y + 5);
                        m_bHasClicked = true;
                        m_rectLab.setm_bHasClicked(true);
                        m_rectLhc.setm_bHasClicked(true);
                        m_rectLhc.setPointMark(point);
                        m_clrMarkLhc = new Color(255-RGB[0], 255-RGB[1],255-RGB[2]);
                        m_rectLhc.setClrMark( m_clrMarkLhc );
                        m_rectLhc.repaint();

                        //force to draw the mark
//                      InvalidateRect(&rectOld);
//                      InvalidateRect(&rect);
```

```
//                  CClientDC dc(this);

                    //get the mouse clicked color
//                  m_clrSelected = RGB(RGB[0], RGB[1], RGB[2]);
                    m_clrSelected = new Color(RGB[0], RGB[1], RGB[2]);

                    m_lhcSelected[0] = m_colorLhc[i][j].Get_L();
                    m_lhcSelected[1] = m_colorLhc[i][j].Get_h();
                    m_lhcSelected[2] = m_colorLhc[i][j].Get_C();

//                  InvalidateRect(&m_rectSample);
                    m_rectSample.setBackground( m_clrSelected );


                    //display the result of value in different color space
                    double XYZ[] = new double[3];
                    double xy[] = new double[2];
                    m_colorLhc[i][j].Get_XYZ(XYZ);
                    for(k = 0; k < 3; k++)
                    {
                            XYZ[k] = ((int)(XYZ[k] * 10000)) / 10000.0;
                    }
                    //calculate of x and y from XYZ
                    cm_XYZ2xy(XYZ[0],XYZ[1],XYZ[2],xy);

                    xy[0] = ((int)(xy[0] * 10000)) / 10000.0;
                    xy[1] = ((int)(xy[1] * 10000)) / 10000.0;

                    m_X = XYZ[0];
                    m_Y = XYZ[1];
                    m_Z = XYZ[2];

                    m_lx = xy[0];
                    m_ly = xy[1];

                    m_red = (int)(rgb[0]);
                    m_green = (int)(rgb[1]);
                    m_blue = (int)(rgb[2]);

                    m_l = m_colorLhc[i][j].Get_L();
                    m_h = m_colorLhc[i][j].Get_h();
                    m_c = m_colorLhc[i][j].Get_C();
                    m_a = m_colorLhc[i][j].Get_a();
                    m_b = m_colorLhc[i][j].Get_b();
                    m_a = ((int)(m_a) * 10000) / 10000.0;
                    m_b = ((int)(m_b) * 10000) / 10000.0;
                    m_h = ((int)(m_h) * 10000) / 10000.0;

                    //color emotion
                    CalculateColorEmotion(m_l, m_h, m_c,true);

                    //slider control position setting
                    m_sldL.setValue(LENGTH_LHC - (int)(m_l+0.5));
                    m_sldC.setValue((int)(m_c+0.5));

                    UpdateData(false);
```

```
                        //for draw the mark point of Lab
                        m_sldA.setValue((int)(m_a+0.5));
                        m_sldB.setValue(-(int)(m_b+0.5));

                        int cx = m_rectLab.getWidth() /2;
                        int cy = m_rectLab.getHeight() /2;
                        m_ptLab = new Point((int)(m_a) + cx, -(int)(m_b) + cy);
                        m_rectLab.setPointMark(m_ptLab);
                        m_clrMarkLab = new Color(255-RGB[0], 255-RGB[1],255-RGB[2]);
                        m_rectLab.setClrMark( m_clrMarkLab );
                        DrawLabGraph();
//                      InvalidateRect(&m_rectLab);
//*/
                }
        }


        void OnInitDialog(){
                GlobalVar.calibrate();
                m_nWebs = 0;


                DrawLabGraph();
                DrawLhcGraph(m_sldH.getValue());

                m_bHasClicked = false;
        }

        void DrawLhcGraph(double h){  //draw the Lhc Graph
                int i,j;
                int rgb[] = new int[3];
                int RGB[] = new int[3];

                //Graphics dc = m_rectLhc.getGraphics();
                //if (dc == null){return;}
                m_colorLhc = m_rectLhc.getColorSpace();

                for(i = 0; i < LENGTH_LHC_C; i++)
                {
                        for(j = 0; j < LENGTH_LHC; j++)
                        {
                                m_colorLhc[i][j].Inilize();
                                m_colorLhc[i][j].Set_C(i);
                                m_colorLhc[i][j].Set_h(h);

                                m_colorLhc[i][j].Set_L(j);
                                m_colorLhc[i][j].ColorLch2XYZ();
                                m_colorLhc[i][j].ColorXYZ2rgb();
                                m_colorLhc[i][j].ColorXYZ2LabCh();
                                m_colorLhc[i][j].Get_rgb(rgb);
                                //repeat the setting of Lch to avoid the deviation of Lhc's
calculated from ColorXYZ2LabCh();
                                m_colorLhc[i][j].Set_C(i);
                                m_colorLhc[i][j].Set_h(h);
                                m_colorLhc[i][j].Set_L(j);
                                for(int k = 0; k < 3; k++)
                                {
                                        RGB[k] = (int)(rgb[k]);
```

```
                                }
                                //is the color can be displayed?
                                if(m_colorLhc[i][j].IsOutOfRange() || max(RGB[0], max(RGB
[1], RGB[2])) > 255 || min(RGB[0], min(RGB[1], RGB[2])) < 0)
                                {
                                        RGB[0] = RGB[1] = RGB[2] = 192;
                                }
                                //draw the color in rectangle 2 x 2

                                //dc.setColor( new Color(RGB[0], RGB[1], RGB[2]) );
                                //HGDIOBJ oldPen = SelectObject(dc, CreatePen
(PS_SOLID, 1, RGB(RGB[0], RGB[1], RGB[2])));
                                //HGDIOBJ oldBrush = SelectObject(dc, brush);
                                //dc.Rectangle((int)(ratio_c*i+0.5), 2*LENGTH_LHC-2*j, (int)
(ratio_c*(i+1)+0.5),2*LENGTH_LHC-(2*j+2));
                                //dc.drawRect((int)(ratio_c*i+0.5), 2*LENGTH_LHC-2*j, (int)
(ratio_c*(i+1)+0.5),2*LENGTH_LHC-(2*j+2));
                                //dc.fillRect((int)(ratio_c*i+0.5), 2*LENGTH_LHC-2*j, 2,2);
                        }
                }
                m_rectLhc.repaint();


        }
        void DrawLabGraph(){                    //draw the Lab Graph
                int i, j, k;
                int rgb[] = new int[3];
                int RGB[] = new int[3];
                int cx, cy;

                //center of the CDC (in memory, not in window client)
                cx = LENGTH_LAB;
                cy = LENGTH_LAB;

                //CBrush brush;
                //brush.CreateSolidBrush(RGB(192, 192, 192));
                //HGDIOBJ oldPen = SelectObject(dc, CreatePen(PS_SOLID, 1, RGB
(192,192,192)));
                //HGDIOBJ oldBrush = SelectObject(dc, brush);
                //CRect rect = CRect(0,0,m_rectLab.Width(), m_rectLab.Height());
                //dc.Rectangle(rect);
                //DeleteObject(SelectObject(dc, oldPen));
                //DeleteObject(SelectObject(dc, oldBrush));

                //Graphics dc = m_rectLab.getGraphics();
                //if (dc == null){return;}

                //m_rectLab.setBackground(bgcolor);

                m_colorLab = m_rectLab.getColorSpace();

                for(i = -LENGTH_LAB; i < LENGTH_LAB; i++)
                {
                        for(j = -LENGTH_LAB; j < LENGTH_LAB; j++)
                        {
                                //calculate the conversion of color space
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Inilize();
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_a(i);
```

```
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_b(j);
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_L
(LENGTH_LHC - m_sldL.getValue());
                                m_colorLab[i+LENGTH_LAB]
[j+LENGTH_LAB].ColorLab2XYZ();
                                m_colorLab[i+LENGTH_LAB]
[j+LENGTH_LAB].ColorXYZ2rgb();
                                m_colorLab[i+LENGTH_LAB]
[j+LENGTH_LAB].ColorXYZ2LabCh();
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_rgb
(rgb);
                                //System.out.println( rgb[0] + " " + rgb[1] + " " + rgb[2]);
                                //repeat the setting of Lab to avoid the deviation of Lhc's
calculated from ColorXYZ2LabCh();
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_a(i);
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_b(j);
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_L
(LENGTH_LHC - m_sldL.getValue());
                                for(k = 0; k < 3; k++)
                                {
                                        RGB[k] = (int)(rgb[k]  + 0.5);
                                }
                                //System.out.println( RGB[0] + " " + RGB[1] + " " + RGB[2]);
                                //System.out.print( m_colorLab[i+LENGTH_LAB]
[j+LENGTH_LAB].IsOutOfRange() +" ");

                                //determine whether the color can be displayed in format
RGB
                                if(m_colorLab[i+LENGTH_LAB]
[j+LENGTH_LAB].IsOutOfRange() || max(RGB[0], max(RGB[1], RGB[2])) > 255 || min(RGB
[0], min(RGB[1], RGB[2])) < 0)
                                {
                                        RGB[0] = RGB[1] = RGB[2] = 192;  //background
color
                                }
                                //draw pixel color, -j+cy stands for axis conversion,from
bottom to top
                                //dc.SetPixel(i+cx, -j+cy, RGB(RGB[0], RGB[1], RGB[2]));
                                //dc.setColor( new Color(RGB[0], RGB[1], RGB[2]) );
                                //dc.setColor( new Color(100+i, 100+j,100+(i+j)/2) );
                                //dc.drawLine( i+cx, -j+cy,  i+cx, -j+cy);
                        }
                }
                m_rectLab.repaint();
        }
        static int max(int a, int b){
                return ( a > b )? a : b;
        }
        static int min(int a, int b){
                return ( a < b )? a : b;
        }

        JPanel contentPane;
        BorderLayout borderLayout1 = new BorderLayout();


        //rectangle area of painting Lhc and Lab space
        hcCanvas m_rectLhc;
```

```
        abCanvas m_rectLab;

        //the mouse click to get the sample
        //is mouse clicked? to avoid the first OnPaint() to display the error color
        boolean m_bHasClicked;
        Canvas m_rectSample;  //rect for painting the sample color
        WebCanvas m_rectWeb;    //rect for the color samples displaying
        int m_nWebs;        //quantities of the  colors sample

        CColorSpace m_colorLhc[][];  //100*100 colors for space Lhc
        CColorSpace m_colorLab[][];  //100*100 colors for space Lab

//      CDC   m_LhcDC, m_LabDC;   //the CDCs of Lhc and Lab painting
//      CBitmap m_bmLhc, m_bmLab;  //the bitmap of Lhc and Lab, related to m_LhcDC and
m_LabDC

        Point m_ptLhc, m_ptLhcOld;  //the point to display the mouse click mark
        Point m_ptLab, m_ptLabOld;

        Color m_clrMarkLhc;      //the color of the mark
        Color m_clrMarkLab;


        //the m_clrSample and m_lhcSample stand the same color. The reason for use two
color
        //space is that the RGB format is in integrate which will cause the error of Hue, and
thus
        //the whole color sample. It is used in mouse left button double click event
        Color m_clrSample[][] = new Color[ROW_SAMPLE][COL_SAMPLE];     //color in
RGB format for saving
        double   m_lhcSample[][][] = new double[ROW_SAMPLE][COL_SAMPLE][3];  //color
in Lhc format for saving
        Color m_clrSelected;                                                    //color
selected
        double   m_lhcSelected[] = new double[3];

        //CStatic         m_ctlWeb;
        //CStatic         m_ctlSample;
        JSlider  m_sldL;
        JSlider  m_sldH;
        JSlider  m_sldC;
        JSlider  m_sldB;
        JSlider  m_sldA;
        //CStatic         m_ctlLhc;
        //CStatic         m_ctlLab;
        //UINT  m_blue;
        //UINT  m_green;
        //UINT  m_red;
        static int         m_blue;
        static int         m_green;
        static int         m_red;
        static double      m_a;
        static double      m_b;
        static double      m_c;
        static double      m_h;
        static double      m_l;
        static double      m_lx;
        static double      m_ly;
```

```
        static double   m_X;
        static double   m_Y;
        static double   m_Z;
        static double   m_deep;
        static double   m_distinct;
        static double   m_dynamic;
        static double   m_gaudy;
        static double   m_heavy;
        static double   m_light;
        static double   m_soft;
        static double   m_striking;
        static double   m_strong;
        static double   m_transparent;
        static double   m_vivid;
        static double   m_warm;


/**Component initialization*/
        private void jbInit() throws Exception  {
                contentPane = (JPanel) this.getContentPane();
                //contentPane.setLayout(borderLayout1);
                contentPane.setLayout(null);
                setSize( FRAME_WIDTH, FRAME_HEIGHT );
                setResizable( false );
                setTitle( "Colour Emotion of Thai");
                setLocation( FRAME_X_ORIGIN, FRAME_Y_ORIGIN );
                addWindowListener( this );
                //addMouseMotionListener( this );
                setBackground( new Color(0xCC, 0xCC, 0xCC));

                exitButton = new Button( "EXIT" );
                exitButton.setFont(new Font("TimesRoman",0,10));
                exitButton.setBounds( 540,515, BUTTON_WIDTH, BUTTON_HEIGHT );
                contentPane.add( exitButton );
                exitButton.addActionListener( this );
                saveButton = new Button( "SAVE" );
                saveButton.setFont(new Font("TimesRoman",0,10));
                saveButton.setBounds( 400, 240,110,30);
                contentPane.add( saveButton );
                saveButton.addActionListener( this );

                textFieldEmotion = new TextField[12];
                labelEmotion = new Label[12];
                for(int i=0; i<textFieldEmotion.length; i++){
                        textFieldEmotion[i] = new TextField("0");
                        textFieldEmotion[i].setBounds( 106+200*(i/6), 30*(i%6+1), 60, 20);
                        textFieldEmotion[i].setFont( new Font("TimesRoman",0,12));
                        textFieldEmotion[i].setBackground( new Color(0xcc, 0xcc, 0xcc) );
                        textFieldEmotion[i].setEnabled( false );
                        //textFieldEmotion[i].setFont( textFont );
                        contentPane.add(textFieldEmotion[i]);
                        labelEmotion[i] = new Label( stringEmotion[i]+":" );
                        labelEmotion[i].setFont( new Font("TimesRoman",0,14));
                        labelEmotion[i].setBounds( 20+165*(i/6), 30*(i%6+1), 120, 20);
                        contentPane.add( labelEmotion[i] );
                }

                labelCValue = new Label[13];
```

```
textFieldCValue = new TextField[13];
for(int i=0; i<13; i++){
        labelCValue[i] = new Label( stringCValue[i] + ":" );
        labelCValue[i].setFont( new Font("TimesRoman",0,12));
        labelCValue[i].setBounds(550, 30+ 22*i, 15, 20);
        contentPane.add( labelCValue[i]);
        textFieldCValue[i] = new TextField( "0" );
        textFieldCValue[i].setBounds(570, 30+ 22*i, 100, 20);
        textFieldCValue[i].setFont( new Font("TimesRoman",0,12));
        textFieldCValue[i].setBackground( new Color( 0xcc,0xcc, 0xcc) );
        textFieldCValue[i].setFont( new Font("TimesRoman",0,12));
        contentPane.add( textFieldCValue[i] );
}

//JSlider(int orientation, int min, int max, int value)

Label labelH = new Label("Hue");
labelH.setBounds(160,230,50,20);
//labelH.setBackground( Color.blue );
labelH.setFont( new Font("TimesRoman",0,14));
contentPane.add( labelH );
Label labelL = new Label("L*");
labelL.setBounds(30,390,20,20);
labelL.setFont( new Font("TimesRoman",0,14));
contentPane.add( labelL );
Label labelC = new Label("C*");
labelC.setBounds(160,520,20,20);
labelC.setFont( new Font("TimesRoman",0,14));
contentPane.add( labelC );
Label labela = new Label("a*");
labela.setBounds(420,520,20,20);
labela.setFont( new Font("TimesRoman",0,14));
contentPane.add( labela );
Label labelb = new Label("b*");
labelb.setBounds(280,390,20,20);
labelb.setFont( new Font("TimesRoman",0,14));
contentPane.add( labelb );

m_sldH = new JSlider(JSlider.HORIZONTAL, 0, 360,0);
//m_sldH = new Scrollbar(Scrollbar.HORIZONTAL, 0, 1, 0, 360);
//m_sldH.setBlockIncrement(10);
//m_sldH.setMajorTickSpacing(36);
//m_sldH.setMinorTickSpacing(36);
//m_sldH.setSnapToTicks(true);
m_sldH.setBounds( 75, 260, 200, 50 );
contentPane.add( m_sldH );
m_sldH.addChangeListener( this );
//m_sldH.addMouseMotionListener( this );
m_sldH.addMouseListener( this );
m_sldH.setLabelTable(m_sldH.createStandardLabels(90));
m_sldH.setPaintLabels(true);

m_sldL = new JSlider(JSlider.VERTICAL, 0, 100, LENGTH_LHC-60);
//m_sldL.setBlockIncrement(10);
m_sldL.setInverted(true);
m_sldL.setBounds( 50, 300, 50, 200 );
contentPane.add( m_sldL );
m_sldL.addChangeListener( this );
```

```
m_sldL.addMouseListener( this );
Hashtable sldLabels = new Hashtable();
sldLabels.put(new Integer(100),new JLabel("0"));
sldLabels.put(new Integer(80),new JLabel("20"));
sldLabels.put(new Integer(60),new JLabel("40"));
sldLabels.put(new Integer(40),new JLabel("60"));
sldLabels.put(new Integer(20),new JLabel("80"));
sldLabels.put(new Integer(0),new JLabel("100"));
m_sldL.setLabelTable(sldLabels);
m_sldL.setPaintLabels(true);

m_sldA = new JSlider(JSlider.HORIZONTAL, -100, 100, 0);
m_sldA.setBounds( 330, 500, 200, 40 );
//m_sldA.setEnabled( false );
contentPane.add( m_sldA );
m_sldA.setLabelTable(m_sldA.createStandardLabels(40));
m_sldA.setPaintLabels(true);

m_sldB = new JSlider(JSlider.VERTICAL, -100, 100, 0);
m_sldB.setInverted(true);
m_sldB.setBounds( 300, 300, 40, 200 );
contentPane.add( m_sldB );
Hashtable sldBLabels = new Hashtable();
sldBLabels.put(new Integer(100),new JLabel("-100"));
sldBLabels.put(new Integer(75),new JLabel("-75"));
sldBLabels.put(new Integer(50),new JLabel("-50"));
sldBLabels.put(new Integer(25),new JLabel("-25"));
sldBLabels.put(new Integer(0),new JLabel("0"));
sldBLabels.put(new Integer(-100),new JLabel("100"));
sldBLabels.put(new Integer(-75),new JLabel("75"));
sldBLabels.put(new Integer(-50),new JLabel("50"));
sldBLabels.put(new Integer(-25),new JLabel("25"));
m_sldB.setLabelTable(sldBLabels);
m_sldB.setPaintLabels(true);

m_sldC = new JSlider(JSlider.HORIZONTAL, 0, 100, 0);
m_sldC.setBounds( 70, 500, 200, 40 );
contentPane.add( m_sldC );
m_sldC.setLabelTable(m_sldC.createStandardLabels(20));
m_sldC.setPaintLabels(true);

m_rectLhc = new hcCanvas();
m_rectLhc.setBounds(75,300, 200, 200);
m_rectLhc.setBackground( bgcolor );
m_rectLhc.addMouseListener( this );
contentPane.add( m_rectLhc );

m_rectLab = new abCanvas();
m_rectLab.setBounds( 330,300,200,200);
m_rectLab.setBackground( bgcolor );
m_rectLab.addMouseListener( this );
contentPane.add( m_rectLab );

m_rectSample = new Canvas();
m_rectSample.setBounds(400,50 ,120,170);
m_rectSample.setBackground( bgcolor );
contentPane.add( m_rectSample );
```

```
            m_rectWeb = new WebCanvas(ROW_SAMPLE,COL_SAMPLE);
            m_rectWeb.setBounds(550,335,120,160);
            contentPane.add( m_rectWeb );
        }

        public CColorConvertDlg(){
            enableEvents(AWTEvent.WINDOW_EVENT_MASK);
            try {
              jbInit();
            }
            catch(Exception e) {
              e.printStackTrace();
            }

            m_a = 0.0;
            m_b = 0.0;
            m_blue = 0;
            m_c = 0.0;
            m_green = 0;
            m_red = 0;
            m_h = 0.0;
            m_l = 0.0;
            m_lx = 0.0;
            m_ly = 0.0;
            m_X = 0.0;
            m_Y = 0.0;
            m_Z = 0.0;
            m_deep = 0.0;
            m_distinct = 0.0;
            m_dynamic = 0.0;
            m_gaudy = 0.0;
            m_heavy = 0.0;
            m_light = 0.0;
            m_soft = 0.0;
            m_striking = 0.0;
            m_strong = 0.0;
            m_transparent = 0.0;
            m_vivid = 0.0;
            m_warm = 0.0;
            m_ptLhc = new Point(0,0);
            m_ptLab = new Point(0,0);
            m_ptLhcOld = new Point(0,0);
            m_ptLabOld = new Point(0,0);
            OnInitDialog();
        }
/*
        void DrawLhcGraph(CDC& dc, double h);  //draw the Lhc Graph
        void DrawLabGraph(CDC& dc);                    //draw the Lab Graph

        void DrawPointMark(CDC& dc, CPoint point, COLORREF& colorref); //draw the
mouse click mark

        void DrawWebColors(CDC& dc);        //draw the saved color
//*/


        void DrawPointMark(Point point, Color colorref){} //draw the mouse click mark
```

```
        void DrawWebColors(){}          //draw the saved color


        void SavePanelColorAndAttribute(String filename){}
        float DoubleTypeFormat(double number){ return 0.0f;}


        // implements WindowListener
        public void windowClosing( WindowEvent event ){
                System.out.println("Closing");
                m_rectWeb.savePanel();
                //System.exit(0);
        }
        public void windowActivated( WindowEvent event ){};
        public void windowClosed( WindowEvent event ){};
        public void windowDeactivated( WindowEvent event ){};
        public void windowDeiconified( WindowEvent event ){};
        public void windowIconified( WindowEvent event ){};
        public void windowOpened( WindowEvent event ){};

        // implements ActionListener
        public void actionPerformed( ActionEvent event ){
                String label = event.getActionCommand();
                System.out.println( label );
                if (label.equals("SAVE"))
                {
                        //setTitle( "You Clicked SAVE" );
                        if (m_bHasClicked)
                        {
                                m_rectWeb.insertColor( m_rectSample.getBackground
(),m_lhcSelected);
                        }
                } else if (label.equals("EXIT")){
                        m_rectWeb.savePanel();
                        System.exit(0);
                }

        }

        // implements MouseListener
        public void mouseClicked(MouseEvent e){
                int x = e.getX();
                int y = e.getY();
                //setTitle( "CLICKED x=" + x + "y="+ y );
                if (e.getSource() == m_rectLhc)
                {
                        onClickedRectLhc(e.getPoint());
                } else if (e.getSource() == m_rectLab){
                        onClickedRectLab(e.getPoint());
                }

                //DrawLabGraph();
                //DrawLhcGraph(m_sldH.getValue());
        }
        public void mousePressed(MouseEvent e){ }
        public void mouseReleased(MouseEvent e){
                if ( e.getSource() == m_sldH )
```

```
                {
                        m_ptLab = m_ptLhc = new Point(0,0);
                        m_rectLab.setPointMark( m_ptLab );
                        m_rectLhc.setPointMark( m_ptLhc );
                        DrawLhcGraph(m_sldH.getValue());
                        if (m_bHasClicked)
                        {
                                m_rectLab.repaint();
                                m_bHasClicked = false;
                                m_rectLab.setm_bHasClicked(true);
                                m_rectLhc.setm_bHasClicked(true);
                        }
                        System.out.println("m_sldH " + m_sldH.getValue());
                } else if ( e.getSource() == m_sldL )
                {
                        m_ptLab = m_ptLhc = new Point(0,0);
                        m_rectLab.setPointMark( m_ptLab );
                        m_rectLhc.setPointMark( m_ptLhc );
                        DrawLabGraph();
                        if (m_bHasClicked)
                        {
                                m_rectLhc.repaint();
                                m_bHasClicked = false;
                                m_rectLab.setm_bHasClicked(true);
                                m_rectLhc.setm_bHasClicked(true);
                        }
                        System.out.println("m_sldL " + m_sldL.getValue());
                }
        }
        public void mouseEntered(MouseEvent e){ }
        public void mouseExited(MouseEvent e){ }

        // implements MouseMotionListener
        public void mouseDragged(MouseEvent e){
        }
        public void mouseMoved(MouseEvent e){
                setTitle("x="+ e.getX() +" y="+ e.getY());
        }


        static TextField[] textFieldEmotion;
        Label[] labelEmotion;
        static final String[] stringEmotion = { "Warm-Cool", "Light-Dark", "Deep-Pale", "Heavy-
Light",
                                                "Vivid-
Sombre", "Gaudy-Plain", "Striking-Subdued", "Strong-Weak",
                                                "Dynamic-
Passive", "Distinct-Vague", "Transparent-Turbid", "Soft-Hard"
                                                };
        Label[] labelCValue;
        static TextField[] textFieldCValue;
        static final String[] stringCValue = { "R", "G", "B", "L*", "a*", "b*", "C*", "h", "X", "Y", "Z",
"x", "y"};
        Button exitButton;
        Button saveButton;
```

```
/*******************************************************/
/*
 * Transforming the tristimulus values to the
 *            chromaticity coordinates
 *      Input parameters : X, Y, Z (tristimulus values)
 *      Output parametsrs : x, y
 */
void cm_XYZ2xy(double X,double Y,double Z,double xy[])
{
        double  sum;

        sum = X + Y + Z;
        if (sum == 0.0)
        {
                xy[0] = 0.0;
                xy[1] = 0.0;
        }
        else
        {
                xy[0] = X/sum;
                xy[1] = Y/sum;
        }
}

static void UpdateData(boolean flag){
        //static final String[] stringCValue = { "R", "G", "B", "L*", "a*", "b*", "C*", "h",
"X", "Y", "Z", "x", "y"};
        //TextField[] textFieldCValue;
        textFieldCValue[0].setText( String.valueOf(m_red));
        textFieldCValue[1].setText( String.valueOf(m_green));
        textFieldCValue[2].setText( String.valueOf(m_blue));
        textFieldCValue[3].setText( String.valueOf(m_l));
        textFieldCValue[4].setText( String.valueOf(m_a));
        textFieldCValue[5].setText( String.valueOf(m_b));
        textFieldCValue[6].setText( String.valueOf(m_c));
        textFieldCValue[7].setText( String.valueOf(m_h));
        textFieldCValue[8].setText( String.valueOf(m_X));
        textFieldCValue[9].setText( String.valueOf(m_Y));
        textFieldCValue[10].setText( String.valueOf(m_Z));
        textFieldCValue[11].setText( String.valueOf(m_lx));
        textFieldCValue[12].setText( String.valueOf(m_ly));

        /*      static final String[] stringEmotion = { "Warm-Cool", "Light-Dark",
"Deep-Pale", "Heavy-Light",

"Vivid-Sombre", "Gaudy-Plain", "Striking-Subdued", "Strong-Weak",

"Dynamic-Passive", "Distinct-Vague", "Transparent-Turbid", "Soft-Hard"

};
        */
        textFieldEmotion[0].setText( String.valueOf(m_warm));
        textFieldEmotion[1].setText( String.valueOf(m_light));
        textFieldEmotion[2].setText( String.valueOf(m_deep));
        textFieldEmotion[3].setText( String.valueOf(m_heavy));
        textFieldEmotion[4].setText( String.valueOf(m_vivid));
        textFieldEmotion[5].setText( String.valueOf(m_gaudy));
        textFieldEmotion[6].setText( String.valueOf(m_striking));
```

```
            textFieldEmotion[7].setText( String.valueOf(m_strong));
            textFieldEmotion[8].setText( String.valueOf(m_dynamic));
            textFieldEmotion[9].setText( String.valueOf(m_distinct));
            textFieldEmotion[10].setText( String.valueOf(m_transparent));
            textFieldEmotion[11].setText( String.valueOf(m_soft));
        }

        public static void main(String[] args)
        {
            System.out.println("Hello World!");
        }


}
```

APPENDIX B

Program 02

Java applet program

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.applet.*;
import javax.swing.event.*;
import java.util.*;

public class CColorConvertDlg  extends Applet implements ActionListener, WindowListener,
MouseListener, MouseMotionListener, ChangeListener
{
        private static final int FRAME_WIDTH = 700;
        private static final int FRAME_HEIGHT = 575;

        private static final int FRAME_X_ORIGIN = 50;
        private static final int FRAME_Y_ORIGIN = 0;

        private static final int BUTTON_WIDTH = 150;
        private static final int BUTTON_HEIGHT = 30;

        static final int LENGTH_LHC_C = 100;
        static final int LENGTH_LHC   = 100;
        static final int LENGTH_LAB   = 100;

        static final int ROW_SAMPLE   = 6;
        static final int COL_SAMPLE   = 6;

        static final int ratio_c      = 2;//(200.0 / LENGTH_LHC_C)
        static final Color bgcolor = new Color( 0xCC, 0xCC, 0xCC );

        static int m_Hold = 0;
        static int m_Lold = LENGTH_LHC-60;

        void CalculateColorEmotion(double L, double h, double C, boolean bRefresh){

                double  deep;
                double  distinct;
                double  dynamic;
                double  gaudy;
                double  heavy;
                double  light;
                double  soft;
                double  striking;
                double  strong;
                double  transparent;
                double  vivid;
                double  warm;

                double dh40 = 40-h;
                double dh290 = h+70;
//              if ( (0<h) && (h<=110) )   dh290 = h+70;
                if ( (110<h) && (h<=290) ) dh290 = 290 - h;
                if ( (290<h) && (h<=360) ) dh290 = h - 290;

//              if ( (0<h) && (h<=40) )   dh40 = 40 - h;
                if ( (40<h) && (h<=220) ) dh40 = h - 40;
                if ( (220<h) && (h<=360) ) dh40 = 400 - h;
```

```
                [illegible]360) * C;

                light           = Math.pow( Math.pow(3.4*(L-10),2) + Math.pow(4.5
*one_dhC,2), 0.5 ) - 184;
                soft            = - Math.pow( Math.pow(2.2*(L-90),2) + Math.pow(0.9
*one_dhC,2),0.5) + 79;
                warm            = Math.pow( Math.pow(0.27*(L-100),2) + Math.pow(1.48*(1
+Math.cos(Math.toRadians(dh40)))*one_dhC,2),0.5) - 58;
                transparent     = Math.pow( Math.pow(3.1*(L-30),2) + Math.pow(2.7
*one_dhC,2),0.5) - 122;
                deep            = Math.pow( Math.pow(2.6*(L-100),2) + Math.pow(1.8
*one_dhC,2),0.5) - 90;
                distinct = Math.pow( Math.pow(1.9*(L-60),2) + Math.pow(3.3*one_dhC,2),0.5) -
62;
                heavy           = Math.pow( Math.pow(2.6*(L-100),2) + Math.pow(0.6
*one_dhC,2),0.5) - 96;
                vivid           = Math.pow( Math.pow(2.2*(L-10),2) + Math.pow(5
*one_dhC,2),0.5) - 157;
                strong          = Math.pow( Math.pow(2.1*(L-90),2) + Math.pow(0.6
*one_dhC,2),0.5) - 52;
                dynamic                 = Math.pow( Math.pow(1.1*(L-20),2) + Math.pow(3.8
*one_dhC,2),0.5) - 100;
                gaudy           = Math.pow( Math.pow(0.4*(L-10),2) + Math.pow(3.8
*one_dhC,2),0.5) - 95;
                striking = Math.pow( Math.pow(1.6*(L-90),2) + Math.pow(3.1*one_dhC,2),0.5) -
65;


                //double emotion[12] = {0};
                double emotion[] = new double[12];
                emotion[0] = deep;
                emotion[1] = distinct;
                emotion[2] = dynamic;
                emotion[3] = gaudy;
                emotion[4] = heavy;
                emotion[5] = light;
                emotion[6] = soft;
                emotion[7] = striking;
                emotion[8] = strong;
                emotion[9] = transparent;
                emotion[10] = vivid;
                emotion[11] = warm;

                for(int i = 0; i < 12; i++)
                {
                        if(emotion[i] < -100)
                                emotion[i] = -100.0;
                        else if(emotion[i] > 100.0)
                                emotion[i] = 100.0;
                        emotion[i] = (int)(emotion[i] * 100) / 100.0;

                }

                m_deep    = emotion[0];
                m_distinct        = emotion[1];
```

```
                m_dynamic       = emotion[2];
                m_gaudy         = emotion[3];
                m_heavy         = emotion[4];
                m_light     = emotion[5];
                m_soft      = emotion[6];
                m_striking      = emotion[7];
                m_strong        = emotion[8];
                m_transparent =         emotion[9];
                m_vivid     = emotion[10];
                m_warm          = emotion[11];

                if(bRefresh)
                        UpdateData(false);


        }


        public void stateChanged(ChangeEvent e){
                if (e.getSource() == m_sldH)
                {
                                if ( m_Hold > m_sldH.getValue())
                                {
                                        m_Hold = m_sldH.getValue();
                                        m_Hold = m_Hold - (m_Hold % 10);
                                        m_sldH.setValue(m_Hold);
                                } else if( m_Hold < m_sldH.getValue())
                                {
                                        m_Hold = m_sldH.getValue();
                                        m_Hold = m_Hold -( m_Hold % 10) + ( ((m_Hold % 10)
==0)?0:10 );
                                        m_sldH.setValue(m_Hold);
                                }
                                System.out.println( "change" );
                } else if(e.getSource() == m_sldL){
                                if ( m_Lold > m_sldL.getValue())
                                {
                                        m_Lold = m_sldL.getValue();
                                        m_Lold = m_Lold - (m_Lold % 10);
                                        m_sldL.setValue(m_Lold);
                                } else if( m_Lold < m_sldL.getValue())
                                {
                                        m_Lold = m_sldL.getValue();
                                        m_Lold = m_Lold -( m_Lold % 10) + ( ((m_Lold % 10)==
0)?0:10 );
                                        m_sldL.setValue(m_Lold);
                                }
                                System.out.println( "change" );
                }
        }

        void onClickedRectLab(Point point){
                //setTitle("Lab");
                int rgb[]= new int[3];
                int RGB[]= new int[3];
                int i,j,k;
```

```
                int cx, cy;
                cx = LENGTH_LAB;
                cy = LENGTH_LAB;

                i = (point.x - cx);
                j = (point.y - cy);
                j =  - j;

                m_colorLab[LENGTH_LAB + i][LENGTH_LAB+j].Get_rgb(rgb);
                for(k = 0; k < 3; k++)
                {
                        RGB[k] = (int)(rgb[k]);
                }

                if(!m_colorLab[LENGTH_LAB + i][LENGTH_LAB+j].IsOutOfRange() &&
                        max(RGB[0], max(RGB[1], RGB[2])) <= 255 && min(RGB[0], min(RGB
[1], RGB[2])) >= 0)
                {
                        m_ptLabOld = m_ptLab;
                        m_ptLab = point;
//                      rectOld = CRect(m_ptLabOld.x - 4, m_ptLabOld.y - 4, m_ptLabOld.x + 5,
m_ptLabOld.y + 5);
//                      rect    = CRect(m_ptLab.x - 4, m_ptLab.y - 4, m_ptLab.x + 5, m_ptLab.y +
5);
                        m_clrMarkLab = new Color(255-RGB[0], 255-RGB[1],255-RGB[2]);
                        m_bHasClicked = true;
                        m_rectLab.setm_bHasClicked(true);
                        m_rectLhc.setm_bHasClicked(true);
                        m_rectLab.setPointMark(point);
                        m_rectLab.setClrMark( m_clrMarkLab );
                        m_rectLab.repaint();



//                      InvalidateRect(&rectOld);
//                      InvalidateRect(&rect);

//                      CClientDC dc(this);

                        m_clrSelected = new Color(RGB[0], RGB[1], RGB[2]);
                        m_lhcSelected[0] = m_colorLab[LENGTH_LAB + i]
[LENGTH_LAB+j].Get_L();
                        m_lhcSelected[1] = m_colorLab[LENGTH_LAB + i]
[LENGTH_LAB+j].Get_h();
                        m_lhcSelected[2] = m_colorLab[LENGTH_LAB + i]
[LENGTH_LAB+j].Get_C();

                        //InvalidateRect(&m_rectSample);
                        m_rectSample.setBackground( m_clrSelected );

                        double XYZ[] = new double[3];
                        double xy[] = new double[2];
                        m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_XYZ(XYZ);
                        for(k = 0; k < 3; k++)
                        {
                                XYZ[k] = ((int)(XYZ[k] * 10000)) / 10000.0;
```

```
                    }
                    cm_XYZ2xy(XYZ[0],XYZ[1],XYZ[2],xy);

                    xy[0] = ((int)(xy[0] * 10000)) / 10000.0;
                    xy[1] = ((int)(xy[1] * 10000)) / 10000.0;

                    m_X = XYZ[0];
                    m_Y = XYZ[1];
                    m_Z = XYZ[2];

                    m_lx = xy[0];
                    m_ly = xy[1];

                    m_red = (int)(rgb[0]);
                    m_green = (int)(rgb[1]);
                    m_blue = (int)(rgb[2]);

                    m_l = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_L();
                    m_h = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_h();
                    m_c = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_C();
                    m_a = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_a();
                    m_b = m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_b();
                    m_a = ((int)(m_a * 10000)) / 10000.0;
                    m_b = ((int)(m_b * 10000)) / 10000.0;


                    CalculateColorEmotion(m_l, m_h, m_c,true);

                    m_sldA.setValue((int)m_a);
                    m_sldB.setValue(-(int)m_b);


                    //reflesh the space of the Lhc
                    //for draw the mark point of Lab
                    m_sldH.setValue((int)(m_h+0.5));
                    m_sldC.setValue((int)(m_c+0.5));
                    m_sldL.setValue(LENGTH_LHC - (int)(m_l+0.5));

                    m_c = ((int)(m_c * 10000)) / 10000.0;
                    m_h = ((int)(m_h * 10000)) / 10000.0;

                    UpdateData(false);


                    m_ptLhc = new Point((int)(m_c*ratio_c+0.5), (LENGTH_LHC - (int)(m_l+
0.5))*2);
                    m_rectLhc.setPointMark(m_ptLhc);
                    m_clrMarkLhc = new Color(255-RGB[0], 255-RGB[1],255-RGB[2]);
                    m_rectLhc.setClrMark( m_clrMarkLhc );
                    //BeginWaitCursor();
                    DrawLhcGraph(m_h);
                    //dc.BitBlt(m_rectLhc.left, m_rectLhc.top, m_rectLhc.Width(),
m_rectLhc.Height(), &m_LhcDC, 0, 0, SRCCOPY);
                    //EndWaitCursor();
```

```
                        //InvalidateRect(&m_rectLhc);
//*/
                }
        }

        void onClickedRectLhc(Point point){
                int rgb[]= new int[3];
                int RGB[]= new int[3];
                int i,j,k;
                //setTitle("x="+point.getX()+" y="+point.getY());

                //calculate the address, caution is the address of j is in opposite
                i = (int)(point.getX() / (ratio_c) + 0.5);
                j = (int)(point.getY() / 2);
                j = LENGTH_LHC - j;  //reverse the y axis

                //get and display rgb
                m_colorLhc[i][j].Get_rgb(rgb);
                for(k = 0; k < 3; k++)
                {
                        RGB[k] = (int)(rgb[k]);
                }

                if(!m_colorLhc[i][j].IsOutOfRange() && max(RGB[0], max(RGB[1], RGB[2])) <=
255 && min(RGB[0], min(RGB[1], RGB[2])) >= 0)
                {
                        //color and invalide area of mark
//                      m_ptLhcOld = m_ptLhc;
//                      m_ptLhc = point;
//                      rectOld = CRect(m_ptLhcOld.x - 4, m_ptLhcOld.y - 4, m_ptLhcOld.x + 5,
m_ptLhcOld.y + 5);
//                      rect    = CRect(m_ptLhc.x - 4, m_ptLhc.y - 4, m_ptLhc.x + 5, m_ptLhc.y +
5);
                        m_bHasClicked = true;
                        m_rectLab.setm_bHasClicked(true);
                        m_rectLhc.setm_bHasClicked(true);
                        m_rectLhc.setPointMark(point);
                        m_clrMarkLhc = new Color(255-RGB[0], 255-RGB[1],255-RGB[2]);
                        m_rectLhc.setClrMark( m_clrMarkLhc );
                        m_rectLhc.repaint();

                        //force to draw the mark
//                      InvalidateRect(&rectOld);
//                      InvalidateRect(&rect);

//                      CClientDC dc(this);

                        //get the mouse clicked color
//                      m_clrSelected = RGB(RGB[0], RGB[1], RGB[2]);
                        m_clrSelected = new Color(RGB[0], RGB[1], RGB[2]);

                        m_lhcSelected[0] = m_colorLhc[i][j].Get_L();
                        m_lhcSelected[1] = m_colorLhc[i][j].Get_h();
                        m_lhcSelected[2] = m_colorLhc[i][j].Get_C();
```

```
//              InvalidateRect(&m_rectSample);
                m_rectSample.setBackground( m_clrSelected );


                //display the result of value in different color space
                double XYZ[] = new double[3];
                double xy[] = new double[2];
                m_colorLhc[i][j].Get_XYZ(XYZ);
                for(k = 0; k < 3; k++)
                {
                        XYZ[k] = ((int)(XYZ[k] * 10000)) / 10000.0;
                }
                //calculate of x and y from XYZ
                cm_XYZ2xy(XYZ[0],XYZ[1],XYZ[2],xy);

                xy[0] = ((int)(xy[0] * 10000)) / 10000.0;
                xy[1] = ((int)(xy[1] * 10000)) / 10000.0;

                m_X = XYZ[0];
                m_Y = XYZ[1];
                m_Z = XYZ[2];

                m_lx = xy[0];
                m_ly = xy[1];

                m_red = (int)(rgb[0]);
                m_green = (int)(rgb[1]);
                m_blue = (int)(rgb[2]);

                m_l = m_colorLhc[i][j].Get_L();
                m_h = m_colorLhc[i][j].Get_h();
                m_c = m_colorLhc[i][j].Get_C();
                m_a = m_colorLhc[i][j].Get_a();
                m_b = m_colorLhc[i][j].Get_b();
                m_a = ((int)(m_a) * 10000) / 10000.0;
                m_b = ((int)(m_b) * 10000) / 10000.0;
                m_h = ((int)(m_h) * 10000) / 10000.0;

                //color emotion
                CalculateColorEmotion(m_l, m_h, m_c,true);

                //slider control position setting
                m_sldL.setValue(LENGTH_LHC - (int)(m_l+0.5));
                m_sldC.setValue((int)(m_c+0.5));

                UpdateData(false);

                //for draw the mark point of Lab
                m_sldA.setValue((int)(m_a+0.5));
                m_sldB.setValue(-(int)(m_b+0.5));

                int cx = m_rectLab.getWidth() /2;
                int cy = m_rectLab.getHeight() /2;
                m_ptLab = new Point((int)(m_a) + cx, -(int)(m_b) + cy);
                m_rectLab.setPointMark(m_ptLab);
```

```
                    m_clrMarkLab = new Color(255-RGB[0], 255-RGB[1],255-RGB[2]);
                    m_rectLab.setClrMark( m_clrMarkLab );
                    DrawLabGraph();
//                  InvalidateRect(&m_rectLab);
//*/
                }
        }


        void OnInitDialog(){
                GlobalVar.calibrate();
                m_nWebs = 0;


                DrawLabGraph();
                DrawLhcGraph(m_sldH.getValue());

                m_bHasClicked = false;
        }

        void DrawLhcGraph(double h){  //draw the Lhc Graph
                int i,j;
                int rgb[] = new int[3];
                int RGB[] = new int[3];

                //Graphics dc = m_rectLhc.getGraphics();
                //if (dc == null){return;}
                m_colorLhc = m_rectLhc.getColorSpace();

                for(i = 0; i < LENGTH_LHC_C; i++)
                {
                        for(j = 0; j < LENGTH_LHC; j++)
                        {
                                m_colorLhc[i][j].Inilize();
                                m_colorLhc[i][j].Set_C(i);
                                m_colorLhc[i][j].Set_h(h);

                                m_colorLhc[i][j].Set_L(j);
                                m_colorLhc[i][j].ColorLch2XYZ();
                                m_colorLhc[i][j].ColorXYZ2rgb();
                                m_colorLhc[i][j].ColorXYZ2LabCh();
                                m_colorLhc[i][j].Get_rgb(rgb);
                                //repeat the setting of Lch to avoid the deviation of Lhc's
calculated from ColorXYZ2LabCh();
                                m_colorLhc[i][j].Set_C(i);
                                m_colorLhc[i][j].Set_h(h);
                                m_colorLhc[i][j].Set_L(j);
                                for(int k = 0; k < 3; k++)
                                {
                                        RGB[k] = (int)(rgb[k]);
                                }
                                //is the color can be displayed?
                                if(m_colorLhc[i][j].IsOutOfRange() || max(RGB[0], max(RGB[1],
RGB[2])) > 255 || min(RGB[0], min(RGB[1], RGB[2])) < 0)
                                {
```

```
                                        [redacted]= 192;
                                }
                                //draw the color in rectangle 2 x 2

                                //dc.setColor( new Color(RGB[0], RGB[1], RGB[2]) );
                                //HGDIOBJ oldPen = SelectObject(dc, CreatePen(PS_SOLID, 1,
RGB(RGB[0], RGB[1], RGB[2])));
                                //HGDIOBJ oldBrush = SelectObject(dc, brush);
                                //dc.Rectangle((int)(ratio_c*i+0.5), 2*LENGTH_LHC-2*j, (int)
(ratio_c*(i+1)+0.5),2*LENGTH_LHC-(2*j+2));
                                //dc.drawRect((int)(ratio_c*i+0.5), 2*LENGTH_LHC-2*j, (int)
(ratio_c*(i+1)+0.5),2*LENGTH_LHC-(2*j+2));
                                //dc.fillRect((int)(ratio_c*i+0.5), 2*LENGTH_LHC-2*j, 2,2);
                        }
                }
                m_rectLhc.repaint();


        }
        void DrawLabGraph(){                    //draw the Lab Graph
                int i, j, k;
                int rgb[] = new int[3];
                int RGB[] = new int[3];
                int cx, cy;

                //center of the CDC (in memory, not in window client)
                cx = LENGTH_LAB;
                cy = LENGTH_LAB;

                //CBrush brush;
                //brush.CreateSolidBrush(RGB(192, 192, 192));
                //HGDIOBJ oldPen = SelectObject(dc, CreatePen(PS_SOLID, 1, RGB
(192,192,192)));
                //HGDIOBJ oldBrush = SelectObject(dc, brush);
                //CRect rect = CRect(0,0,m_rectLab.Width(), m_rectLab.Height());
                //dc.Rectangle(rect);
                //DeleteObject(SelectObject(dc, oldPen));
                //DeleteObject(SelectObject(dc, oldBrush));

                //Graphics dc = m_rectLab.getGraphics();
                //if (dc == null){return;}

                //m_rectLab.setBackground(bgcolor);

                m_colorLab = m_rectLab.getColorSpace();

                for(i = -LENGTH_LAB; i < LENGTH_LAB; i++)
                {
                        for(j = -LENGTH_LAB; j < LENGTH_LAB; j++)
                        {
                                //calculate the conversion of color space
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Inilize();
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_a(i);
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_b(j);
                                m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_L
(LENGTH_LHC - m_sldL.getValue());
```

```
                                    [j+LENGTH_LAB].ColorLab2XYZ();
                                    m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].ColorXYZ2rgb();
                                    m_colorLab[i+LENGTH_LAB]
[j+LENGTH_LAB].ColorXYZ2LabCh();
                                    m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Get_rgb(rgb);
                                    //System.out.println( rgb[0] + " " + rgb[1] + " " + rgb[2]);
                                    //repeat the setting of Lab to avoid the deviation of Lhc's
calculated from ColorXYZ2LabCh();
                                    m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_a(i);
                                    m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_b(j);
                                    m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].Set_L
(LENGTH_LHC - m_sldL.getValue());
                                    for(k = 0; k < 3; k++)
                                    {
                                            RGB[k] = (int)(rgb[k]  + 0.5);
                                    }
                                    //System.out.println( RGB[0] + " " + RGB[1] + " " + RGB[2]);
                                    //System.out.print( m_colorLab[i+LENGTH_LAB]
[j+LENGTH_LAB].IsOutOfRange() +" ");

                                    //determine whether the color can be displayed in format RGB
                                    if(m_colorLab[i+LENGTH_LAB][j+LENGTH_LAB].IsOutOfRange
() || max(RGB[0], max(RGB[1], RGB[2])) > 255 || min(RGB[0], min(RGB[1], RGB[2])) < 0)
                                    {
                                            RGB[0] = RGB[1] = RGB[2] = 192;  //background color
                                    }
                                    //draw pixel color, -j+cy stands for axis conversion,from bottom to
top
                                    //dc.SetPixel(i+cx, -j+cy, RGB(RGB[0], RGB[1], RGB[2]));
                                    //dc.setColor( new Color(RGB[0], RGB[1], RGB[2]) );
                                    //dc.setColor( new Color(100+i, 100+j,100+(i+j)/2) );
                                    //dc.drawLine( i+cx, -j+cy,  i+cx, -j+cy);
                            }
                    }
                    m_rectLab.repaint();
            }
            static int max(int a, int b){
                    return ( a > b )? a : b;
            }
            static int min(int a, int b){
                    return ( a < b )? a : b;
            }

            JPanel contentPane;
            BorderLayout borderLayout1 = new BorderLayout();


            //rectangle area of painting Lhc and Lab space
            hcCanvas m_rectLhc;
            abCanvas m_rectLab;

            //the mouse click to get the sample
            //is mouse clicked? to avoid the first OnPaint() to display the error color
            boolean m_bHasClicked;
            Canvas m_rectSample;  //rect for painting the sample color
            WebCanvas m_rectWeb;     //rect for the color samples displaying
```

```
sldLabels.put(new Integer(100),new JLabel("0"));
sldLabels.put(new Integer(80),new JLabel("20"));
sldLabels.put(new Integer(60),new JLabel("40"));
sldLabels.put(new Integer(40),new JLabel("60"));
sldLabels.put(new Integer(20),new JLabel("80"));
sldLabels.put(new Integer(0),new JLabel("100"));
m_sldL.setLabelTable(sldLabels);
m_sldL.setPaintLabels(true);

m_sldA = new JSlider(JSlider.HORIZONTAL, -100, 100, 0);
m_sldA.setBounds( 330, 500, 200, 40 );
//m_sldA.setEnabled( false );
contentPane.add( m_sldA );
m_sldA.setLabelTable(m_sldA.createStandardLabels(40));
m_sldA.setPaintLabels(true);

m_sldB = new JSlider(JSlider.VERTICAL, -100, 100, 0);
m_sldB.setInverted(true);
m_sldB.setBounds( 280, 300, 40, 200 );
contentPane.add( m_sldB );
Hashtable sldBLabels = new Hashtable();
sldBLabels.put(new Integer(100),new JLabel("-100"));
sldBLabels.put(new Integer(75),new JLabel("-75"));
sldBLabels.put(new Integer(50),new JLabel("-50"));
sldBLabels.put(new Integer(25),new JLabel("-25"));
sldBLabels.put(new Integer(0),new JLabel("0"));
sldBLabels.put(new Integer(-100),new JLabel("100"));
sldBLabels.put(new Integer(-75),new JLabel("75"));
sldBLabels.put(new Integer(-50),new JLabel("50"));
sldBLabels.put(new Integer(-25),new JLabel("25"));
m_sldB.setLabelTable(sldBLabels);
m_sldB.setPaintLabels(true);

m_sldC = new JSlider(JSlider.HORIZONTAL, 0, 100, 0);
m_sldC.setBounds( 70, 500, 200, 40 );
contentPane.add( m_sldC );
m_sldC.setLabelTable(m_sldC.createStandardLabels(20));
m_sldC.setPaintLabels(true);

m_rectLhc = new hcCanvas();
m_rectLhc.setBounds(75,300, 200, 200);
m_rectLhc.setBackground( bgcolor );
m_rectLhc.addMouseListener( this );
contentPane.add( m_rectLhc );

m_rectLab = new abCanvas();
m_rectLab.setBounds( 330,300,200,200);
m_rectLab.setBackground( bgcolor );
m_rectLab.addMouseListener( this );
contentPane.add( m_rectLab );

m_rectSample = new Canvas();
m_rectSample.setBounds(400,50 ,120,170);
m_rectSample.setBackground( bgcolor );
contentPane.add( m_rectSample );
```

```
            m_rectWeb = new WebCanvas(ROW_SAMPLE,COL_SAMPLE);
            m_rectWeb.setBounds(550,350,120,120);
            contentPane.add( m_rectWeb );

        }

        public void CColorConvertDlg1(){
            enableEvents(AWTEvent.WINDOW_EVENT_MASK);
            try {
              jbInit();
            }
            catch(Exception e) {
              e.printStackTrace();
            }

            m_a = 0.0;
            m_b = 0.0;
            m_blue = 0;
            m_c = 0.0;
            m_green = 0;
            m_red = 0;
            m_h = 0.0;
            m_l = 0.0;
            m_lx = 0.0;
            m_ly = 0.0;
            m_X = 0.0;
            m_Y = 0.0;
            m_Z = 0.0;
            m_deep = 0.0;
            m_distinct = 0.0;
            m_dynamic = 0.0;
            m_gaudy = 0.0;
            m_heavy = 0.0;
            m_light = 0.0;
            m_soft = 0.0;
            m_striking = 0.0;
            m_strong = 0.0;
            m_transparent = 0.0;
            m_vivid = 0.0;
            m_warm = 0.0;
            m_ptLhc = new Point(0,0);
            m_ptLab = new Point(0,0);
            m_ptLhcOld = new Point(0,0);
            m_ptLabOld = new Point(0,0);
            OnInitDialog();
        }

/*
        void DrawLhcGraph(CDC& dc, double h);  //draw the Lhc Graph
        void DrawLabGraph(CDC& dc);                    //draw the Lab Graph

        void DrawPointMark(CDC& dc, CPoint point, COLORREF& colorref); //draw the mouse
click mark

        void DrawWebColors(CDC& dc);        //draw the saved color
//*/
```

```
        void DrawPointMark(Point point, Color colorref){} //draw the mouse click mark

        void DrawWebColors(){}          //draw the saved color


        void SavePanelColorAndAttribute(String filename){}
        float DoubleTypeFormat(double number){ return 0.0f;}


        // implements WindowListener
        public void windowClosing( WindowEvent event ){          System.exit(0);}
        public void windowActivated( WindowEvent event ){};
        public void windowClosed( WindowEvent event ){};
        public void windowDeactivated( WindowEvent event ){};
        public void windowDeiconified( WindowEvent event ){};
        public void windowIconified( WindowEvent event ){};
        public void windowOpened( WindowEvent event ){};

        // implements ActionListener
        public void actionPerformed( ActionEvent event ){
                String label = event.getActionCommand();
                if (label.equals("SAVE"))
                {
                        //setTitle( "You Clicked SAVE" );
                        if (m_bHasClicked)
                        {
                                m_rectWeb.insertColor( m_rectSample.getBackground
(),m_lhcSelected);
                        }
                } else if (label.equals("EXIT")){
                        System.exit(0);
                }

        }

        // implements MouseListener
        public void mouseClicked(MouseEvent e){
                int x = e.getX();
                int y = e.getY();
                //setTitle( "CLICKED x=" + x + "y="+ y );
                if (e.getSource() == m_rectLhc)
                {
                        onClickedRectLhc(e.getPoint());
                } else if (e.getSource() == m_rectLab){
                        onClickedRectLab(e.getPoint());
                }

                //DrawLabGraph();
                //DrawLhcGraph(m_sldH.getValue());
        }
        public void mousePressed(MouseEvent e){ }

        public void mouseReleased(MouseEvent e){
                if ( e.getSource() == m_sldH )
```

```
                {
                        m_ptLab = m_ptLhc = new Point(0,0);
                        m_rectLab.setPointMark( m_ptLab );
                        m_rectLhc.setPointMark( m_ptLhc );
                        DrawLhcGraph(m_sldH.getValue());
                        if (m_bHasClicked)
                        {
                                m_rectLab.repaint();
                                m_bHasClicked = false;
                                m_rectLab.setm_bHasClicked(true);
                                m_rectLhc.setm_bHasClicked(true);
                        }
                        System.out.println("m_sldH " + m_sldH.getValue());
                } else if ( e.getSource() == m_sldL )
                {
                        m_ptLab = m_ptLhc = new Point(0,0);
                        m_rectLab.setPointMark( m_ptLab );
                        m_rectLhc.setPointMark( m_ptLhc );
                        DrawLabGraph();
                        if (m_bHasClicked)
                        {
                                m_rectLhc.repaint();
                                m_bHasClicked = false;
                                m_rectLab.setm_bHasClicked(true);
                                m_rectLhc.setm_bHasClicked(true);
                        }
                        System.out.println("m_sldL " + m_sldL.getValue());
                }
        }

        public void mouseEntered(MouseEvent e){ }
        public void mouseExited(MouseEvent e){ }

        // implements MouseMotionListener
        public void mouseDragged(MouseEvent e){
        }
        public void mouseMoved(MouseEvent e){
                //setTitle("x="+ e.getX() +" y="+ e.getY());
        }


        TextField[] textFieldEmotion;
        Label[] labelEmotion;
        static final String[] stringEmotion = { "Warm-Cool", "Light-Dark", "Deep-Pale", "Heavy-
Light",
                                                                                "Vivid-
Sombre", "Gaudy-Plain", "Striking-Subdued", "Strong-Weak",
                                                                                "Dynamic-
Passive", "Distinct-Vague", "Transparent-Turbid", "Soft-Hard"
                                                                                };
        Label[] labelCValue;
        TextField[] textFieldCValue;
        static final String[] stringCValue = { "R", "G", "B", "L*", "a*", "b*", "C*", "h", "X", "Y", "Z", "x",
"y"};
        Button exitButton;
```

```
/*****************************************************/
/*
 * Transforming the tristimulus values to the
 *            chromaticity coordinates
 *      Input parameters : X, Y, Z (tristimulus values)
 *      Output parametsrs : x, y
 */
void cm_XYZ2xy(double X,double Y,double Z,double xy[])
{
        double  sum;

        sum = X + Y + Z;
        if (sum == 0.0)
        {
                xy[0] = 0.0;
                xy[1] = 0.0;
        }
        else
        {
                xy[0] = X/sum;
                xy[1] = Y/sum;
        }
}

void UpdateData(boolean flag){
        //static final String[] stringCValue = { "R", "G", "B", "L*", "a*", "b*", "C*", "h", "X",
"Y", "Z", "x", "y"};
        //TextField[] textFieldCValue;
        textFieldCValue[0].setText( String.valueOf(m_red));
        textFieldCValue[1].setText( String.valueOf(m_green));
        textFieldCValue[2].setText( String.valueOf(m_blue));
        textFieldCValue[3].setText( String.valueOf(m_l));
        textFieldCValue[4].setText( String.valueOf(m_a));
        textFieldCValue[5].setText( String.valueOf(m_b));
        textFieldCValue[6].setText( String.valueOf(m_c));
        textFieldCValue[7].setText( String.valueOf(m_h));
        textFieldCValue[8].setText( String.valueOf(m_X));
        textFieldCValue[9].setText( String.valueOf(m_Y));
        textFieldCValue[10].setText( String.valueOf(m_Z));
        textFieldCValue[11].setText( String.valueOf(m_lx));
        textFieldCValue[12].setText( String.valueOf(m_ly));

        /*      static final String[] stringEmotion = { "Warm-Cool", "Light-Dark", "Deep-
Pale", "Heavy-Light",

"Vivid-Sombre", "Gaudy-Plain", "Striking-Subdued", "Strong-Weak",

"Dynamic-Passive", "Distinct-Vague", "Transparent-Turbid", "Soft-Hard"

        };
        */
        textFieldEmotion[0].setText( String.valueOf(m_warm));
        textFieldEmotion[1].setText( String.valueOf(m_light));
```

```
                textFieldEmotion[2].setText( String.valueOf(m_deep));
                textFieldEmotion[3].setText( String.valueOf(m_heavy));
                textFieldEmotion[4].setText( String.valueOf(m_vivid));
                textFieldEmotion[5].setText( String.valueOf(m_gaudy));
                textFieldEmotion[6].setText( String.valueOf(m_striking));
                textFieldEmotion[7].setText( String.valueOf(m_strong));
                textFieldEmotion[8].setText( String.valueOf(m_dynamic));
                textFieldEmotion[9].setText( String.valueOf(m_distinct));
                textFieldEmotion[10].setText( String.valueOf(m_transparent));
                textFieldEmotion[11].setText( String.valueOf(m_soft));

        }

        public static void main(String[] args)
        {
                System.out.println("Hello World!");
        }


}
```

APPENDIX C

Program 03

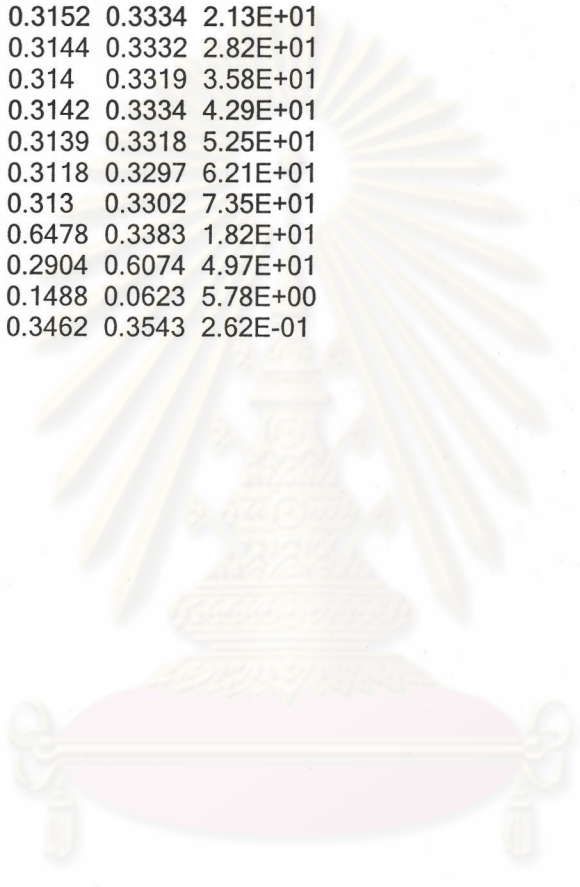
Main program

```
class ColorConvert
{
	public static void main(String[] args)
	{

		//System.out.println("Hello World!");
		if ( args.length > 0)
		{
			GlobalVar.MONITOR_FILE = args[0];
			System.out.println( GlobalVar.MONITOR_FILE );
		}

		CColorConvertDlg dlg = new CColorConvertDlg();
		dlg.setVisible( true );
	}
}
```

APPENDIX D

My monitor.txt

255	255	255	0.313	0.3302	7.35E+01
255	0	0	0.6478	0.3383	1.82E+01
0	255	0	0.2904	0.6074	4.97E+01
0	0	255	0.1488	0.0623	5.78E+00
16	16	16	0.3469	0.3538	2.73E-01
32	32	32	0.3485	0.3498	3.91E-01
48	48	48	0.343	0.348	7.84E-01
64	64	64	0.334	0.3476	1.70E+00
80	80	80	0.3265	0.3421	3.26E+00
96	96	96	0.3228	0.3395	5.29E+00
112	112	112	0.3196	0.3375	8.15E+00
128	128	128	0.3176	0.3356	1.16E+01
144	144	144	0.3161	0.3347	1.62E+01
160	160	160	0.3152	0.3334	2.13E+01
176	176	176	0.3144	0.3332	2.82E+01
192	192	192	0.314	0.3319	3.58E+01
208	208	208	0.3142	0.3334	4.29E+01
224	224	224	0.3139	0.3318	5.25E+01
240	240	240	0.3118	0.3297	6.21E+01
255	255	255	0.313	0.3302	7.35E+01
255	0	0	0.6478	0.3383	1.82E+01
0	255	0	0.2904	0.6074	4.97E+01
0	0	255	0.1488	0.0623	5.78E+00
0	0	0	0.3462	0.3543	2.62E-01



VITA

Miss. Sanichar Koonawoot was born on October 1, 1977 in Angthong, Thailand. She received her B.Sc. degree in Physics from the Facuity of Science, Chulalongkorn University in 1998, and she has been a graduate student in the Imaging Technology Program, Graduate school, Chulalongkorn University since 2002.