画像処理 Image Processing

目標

画像処理のための基礎的な素養と技術を身につける。

計画

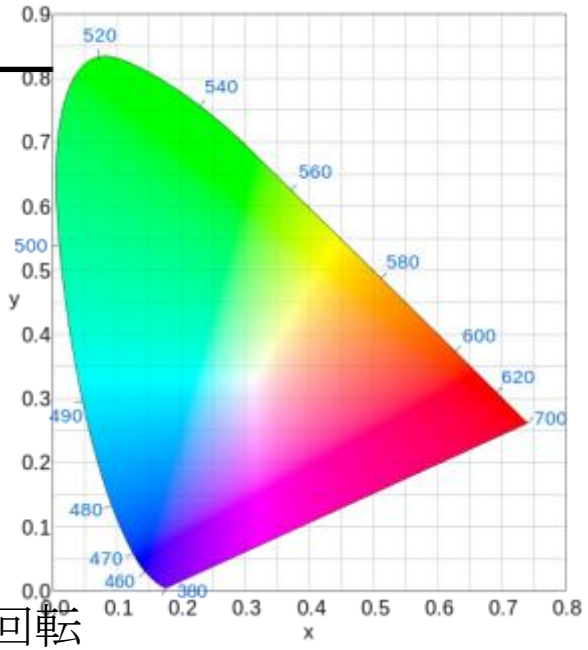
1. プログラミングとEclipse利用環境でのJava
2. 平面図形の表現とデバッグ
3. 色概念とフィルタ処理
4. グレイフィルタ
5. 画像内の部分処理
6. クロマキー入門
7. クロマキー応用
8. 画像の回転
9. 画像の拡大と縮小
10. ぼかし効果とその利用法
11. エッジ抽出と先鋭化
12. 画像処理と保存

参考書とプログラム

根岸利一郎(2016):『ひまわりの黄金比』[日本評論社] の参考プログラム
https://www.nippyo.co.jp/shop/book/7089.html

EclipseやJavaの解説書

1．画像処理入門



画像…表示装置での画素の集合
表示装置…等間隔画素表示
表色…国際照明委員会（Commission internationale de l'éclairage，略称：CIE）の定める表色空間。
RGB（Red，Green，Blue）の加法混色（光）
RGBで表現できない場合ほか，XYZ表色系，マンセル表色系ほか。

画像処理…記録・伝送

- 画像変換
  解像度，明度・彩度，コントラスト，拡大・縮小，回転
- 画像処理効果
  フィルタ処理，グレイスケール処理，ぼかし効果とエッジ抽出
- 画像処理応用
  サンプリング，視認性

内実　…・基礎…数学と物理，一部のみ，ほか省略。
・応用…電子機器とソフトウエア，Javaによる入門。

Java言語とその特徴
1990年代　サン・マイクロシステムズ（Sun Microsystems）で開発

なぜ，Javaが最も使われているか

○オブジェクト指向の徹底…制限が多い
クラス，継承，隠蔽，多態性

○プログラム実行環境に依存しない…JVM（Java仮想マシン，Java Virtual Machine）上で動作，オブジェクト指向コンピュータ ⇒ 移植性が高い

Javaの実行環境 Cとの比較

C言語	Java言語
	ユーザ
ユーザ	Javaプログラムのバイトコード
Cプログラムのマシンコード	Java仮想マシン（JVM）
OS（UNIXやWindows）	OS（UNIXやWindows）
ハードウエア	ハードウエア

C言語	と	Java
・すべて関数		・すべてクラス
・ヘッダファイル(include)		・クラスライブラリ(import)
・関数		・メソッド（サブルーチンのような）
・ポインタ		・アドレス参照
・構造体		・データセット
・実行用.exeファイル		・**.classファイル

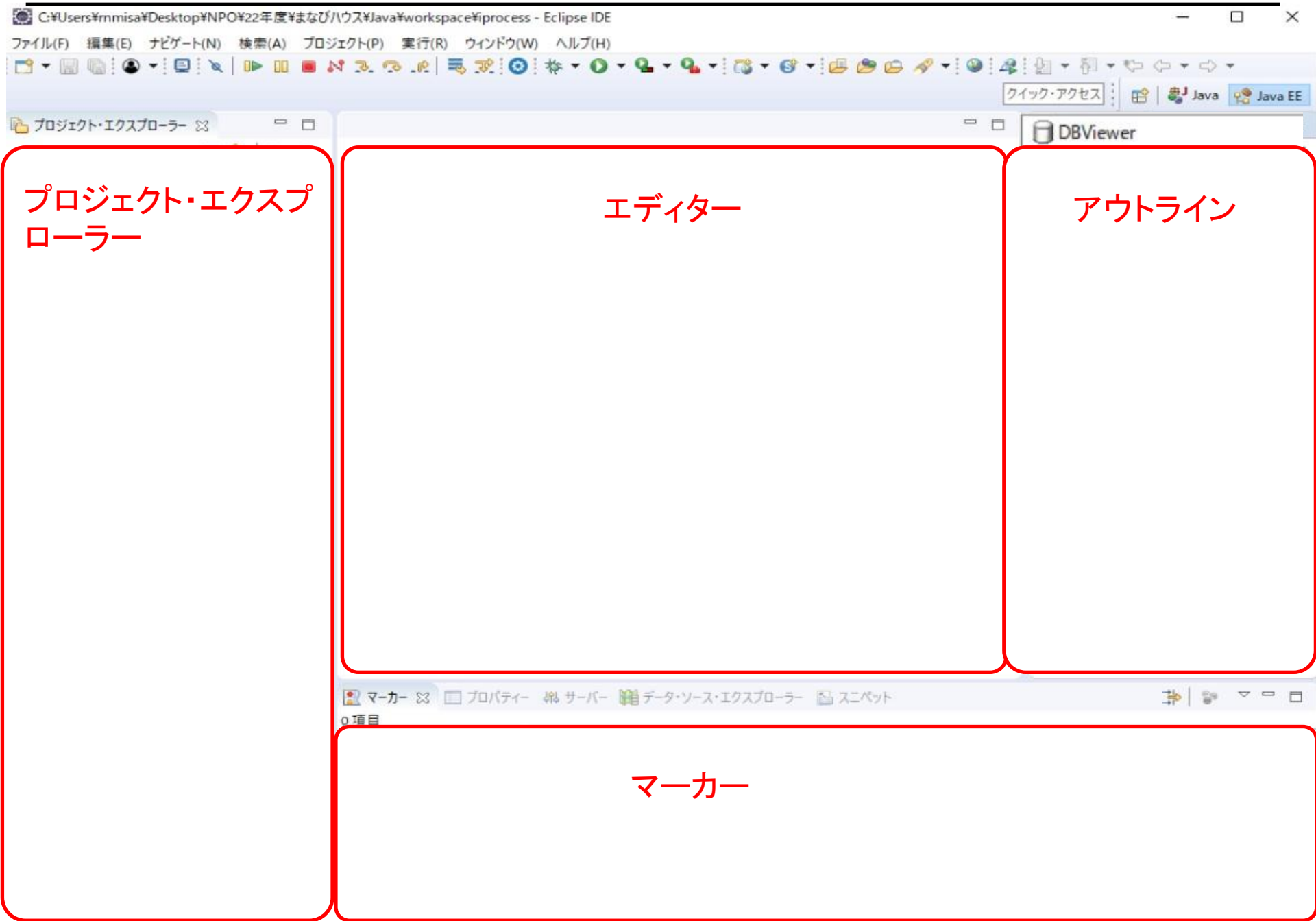
作られるプログラムタイプ

・アプリケーション（Java Application）
・アプレット　　（Java Applet）
・サーブレット　　（Java Servlet）
この実習ではアプリケーション（アプレット）を実習する。

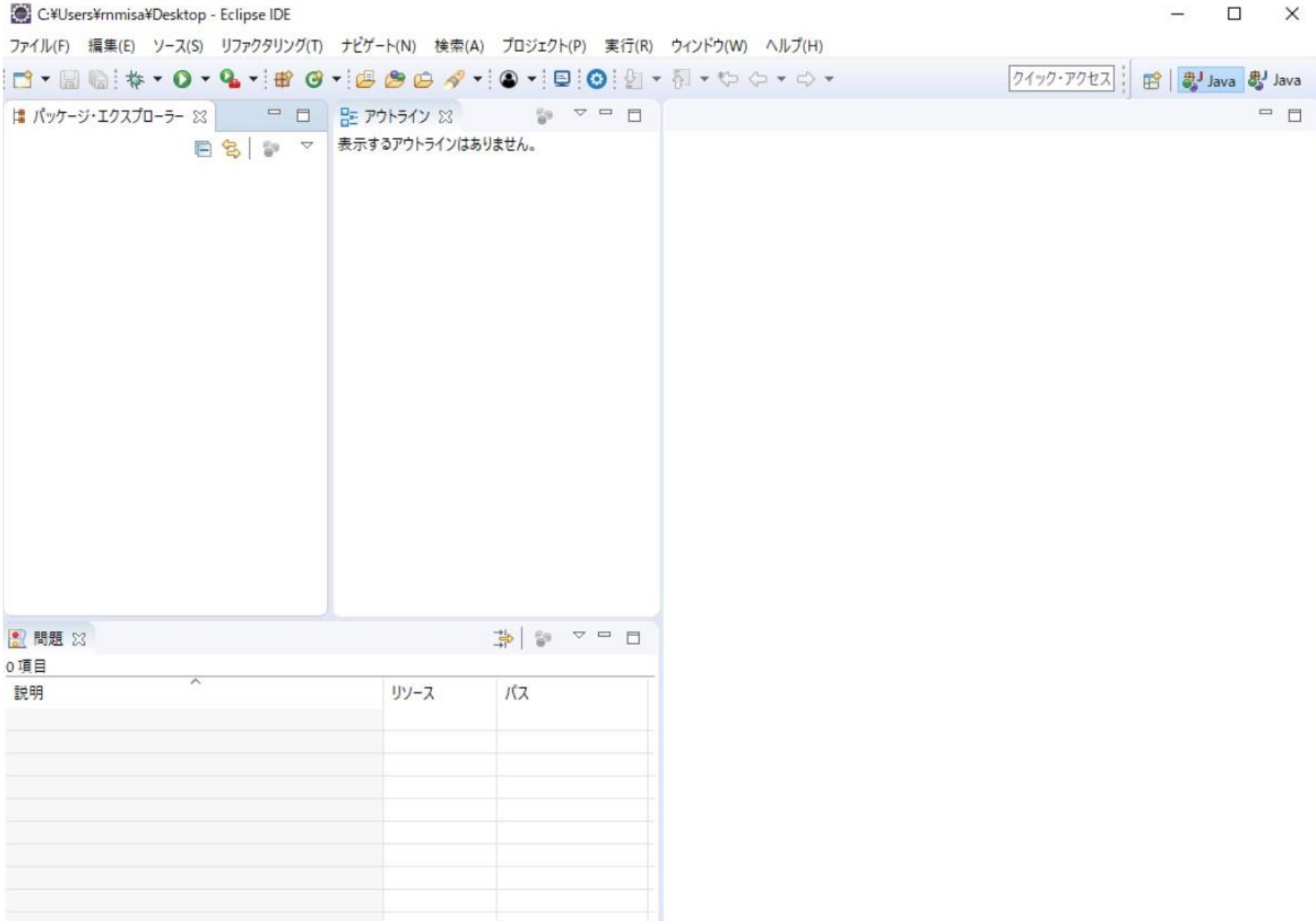
高速処理には向かない。
多量の同時アクセスには対応が困難。

Java 環境
○ Java開発キット：Java　JDK8
○ 開発環境Eclipse 4.＊　2019年からPhotonなどのコード名廃止
「統合開発環境(IDE:Integrated Development Environment)」
○ 個人利用以外の有料化

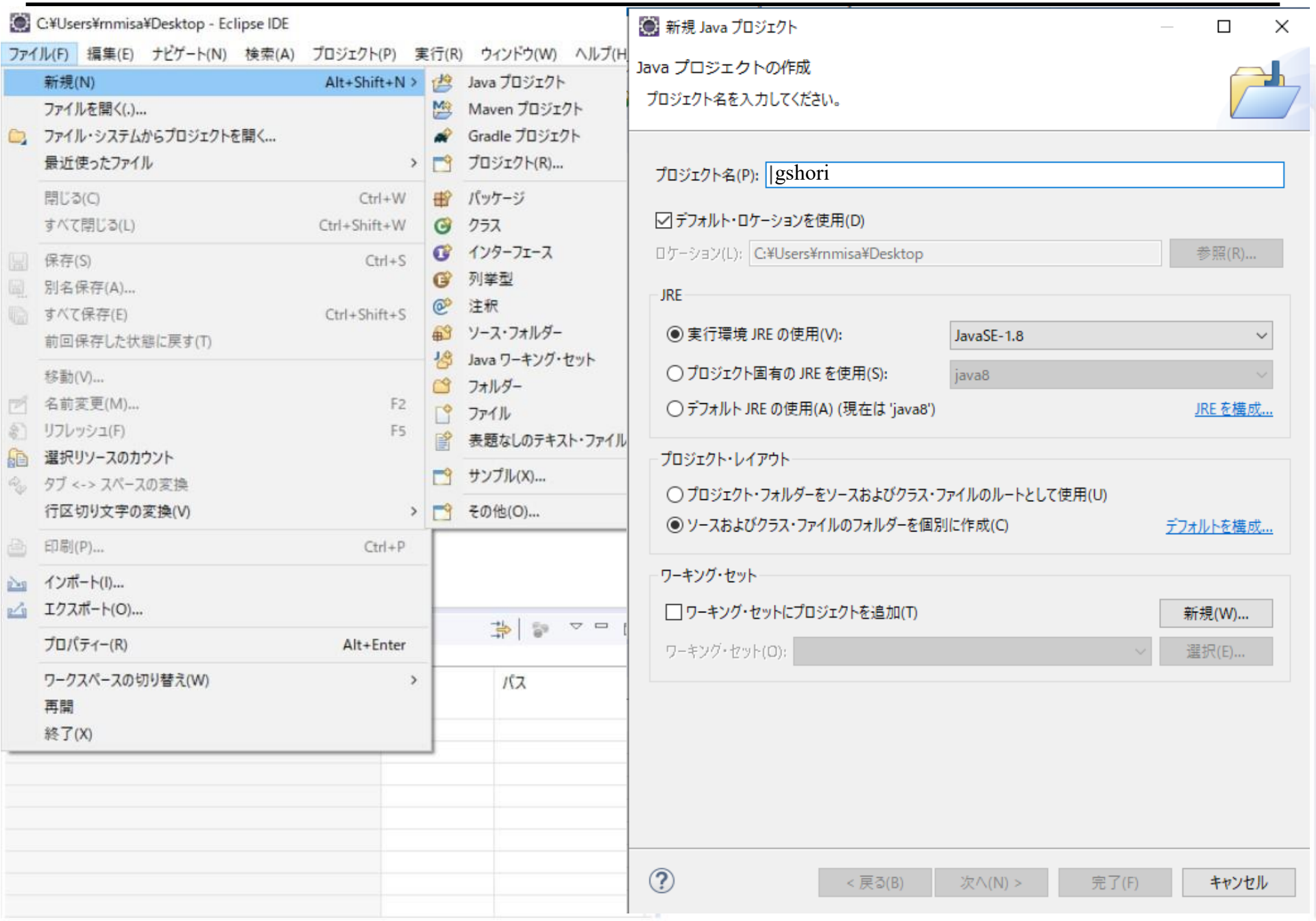
EclipseでJava 環境のワークベンチにする



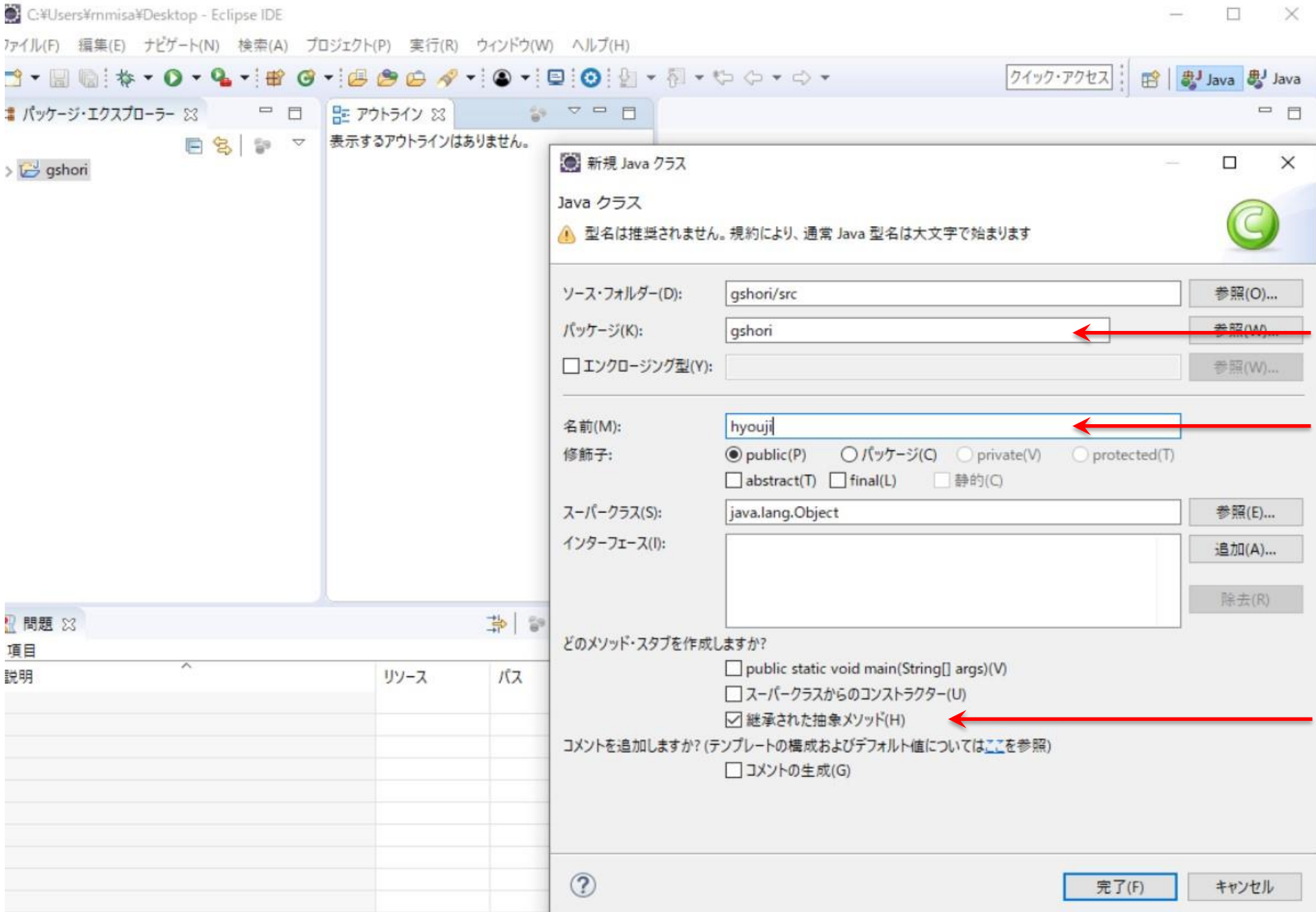
EclipseでJava（デフォルト）環境のワークベンチ



プロジェクトの作成


C:¥Users¥rnmisa¥Desktop - Eclipse IDE
ファイル(F)
編集(E)
ナビゲート(N)
検索(A)
プロジェクト(P)
実行(R)
ウィンドウ(W)
ヘルプ(H
新規(N) Alt+Shift+N
ファイルを開く(.)...
ファイル・システムからプロジェクトを開く...
最近使ったファイル
閉じる(C) Ctrl+W
すべて閉じる(L) Ctrl+Shift+W
保存(S) Ctrl+S
別名保存(A)...
すべて保存(E) Ctrl+Shift+S
前回保存した状態に戻す(T)
移動(V)...
名前変更(M)... F2
リフレッシュ(F) F5
選択リソースのカウント
タブ <-> スペースの変換
行区切り文字の変換(V)
印刷(P)... Ctrl+P
インポート(I)...
エクスポート(O)...
プロパティー(R) Alt+Enter
ワークスペースの切り替え(W)
再開
終了(X)
Java プロジェクト
Maven プロジェクト
Gradle プロジェクト
プロジェクト(R)...
パッケージ
クラス
インターフェース
列挙型
注釈
ソース・フォルダー
Java ワーキング・セット
フォルダー
ファイル
表題なしのテキスト・ファイル
サンプル(X)...
その他(O)...
パス
新規 Java プロジェクト
Java プロジェクトの作成
プロジェクト名を入力してください。
プロジェクト名(P): gshori
デフォルト・ロケーションを使用(D)
ロケーション(L): C:¥Users¥rnmisa¥Desktop
参照(R)...
JRE
実行環境 JRE の使用(V): JavaSE-1.8
プロジェクト固有の JRE を使用(S): java8
デフォルト JRE の使用(A) (現在は 'java8')
JRE を構成...
プロジェクト・レイアウト
プロジェクト・フォルダーをソースおよびクラス・ファイルのルートとして使用(U)
ソースおよびクラス・ファイルのフォルダーを個別に作成(C)
デフォルトを構成...
ワーキング・セット
ワーキング・セットにプロジェクトを追加(T)
新規(W)...
ワーキング・セット(O):
選択(E)...
< 戻る(B)
次へ(N) >
完了(F)
キャンセル

クラスの作成


C:¥Users¥rnmisa¥Desktop - Eclipse IDE
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
クイック・アクセス
Java
Java
パッケージ・エクスプローラー
gshori
アウトライン
表示するアウトラインはありません。
新規 Java クラス
Java クラス
型名は推奨されません。規約により、通常 Java 型名は大文字で始まります
ソース・フォルダー(D):
gshori/src
参照(O)...
パッケージ(K):
gshori
参照(W)...
エンクロージング型(Y):
参照(W)...
名前(M):
hyouji
修飾子:
public(P)
パッケージ(C)
private(V)
protected(T)
abstract(T)
final(L)
静的(C)
スーパークラス(S):
java.lang.Object
参照(E)...
インターフェース(I):
追加(A)...
除去(R)
どのメソッド・スタブを作成しますか?
public static void main(String[] args)(V)
スーパークラスからのコンストラクター(U)
継承された抽象メソッド(H)
コメントを追加しますか? (テンプレートの構成およびデフォルト値についてはここを参照)
コメントの生成(G)
完了(F)
キャンセル
問題
項目
説明
リソース
パス

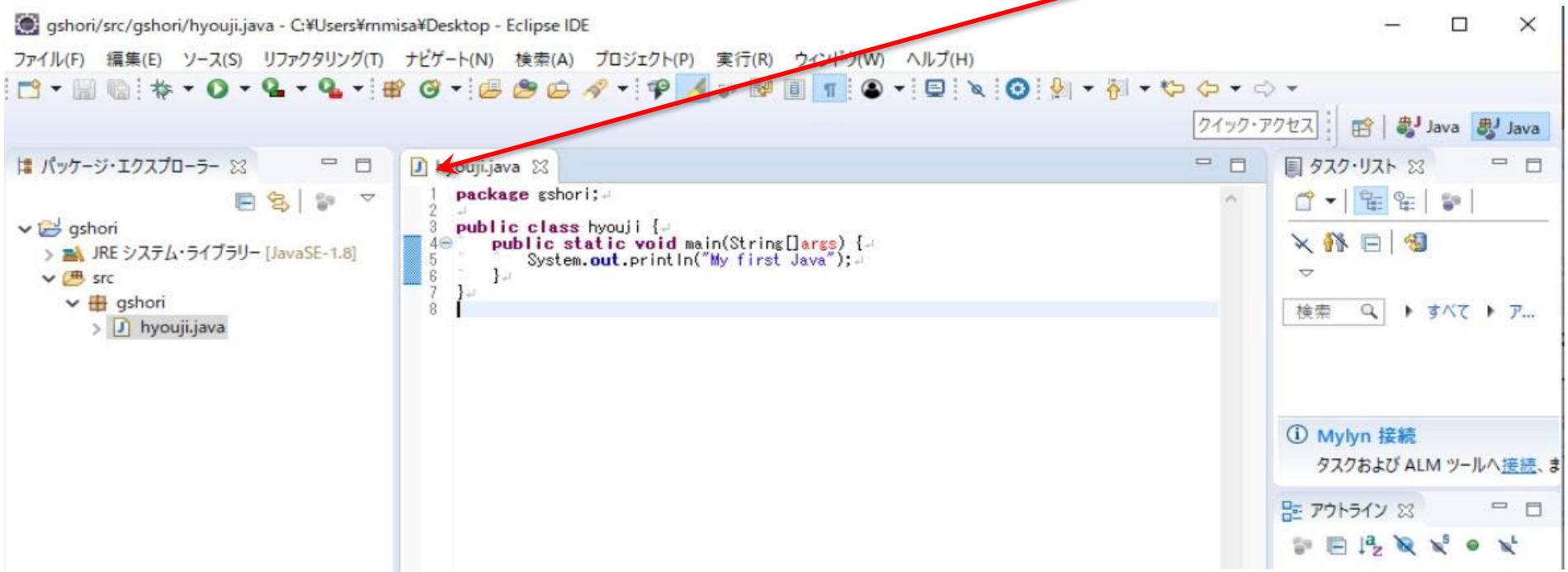
画像表示

```
import java.awt.Graphics;
import java.awt.Image;

public class image extends java.applet.Applet {
    Image logoImage;

   public void init() {
        logoImage = getImage(getCodeBase(), "image/logo.jpg");
    }

   public void paint(Graphics g) {
        g.drawImage(logoImage, 50, 50, this);
        g.drawImage(logoImage, 150, 25, 100, 100, this);
    }
}
```

(配列宣言していない)

最初のアプリケーションプログラムと実行

*:未保存がある



保存環境

USB, Network

//:コメント

コード・アシスト

エラーと表示，クイックフィックス（修正）

保存：Ctrl+S(Mac:command+S),

保存と実行

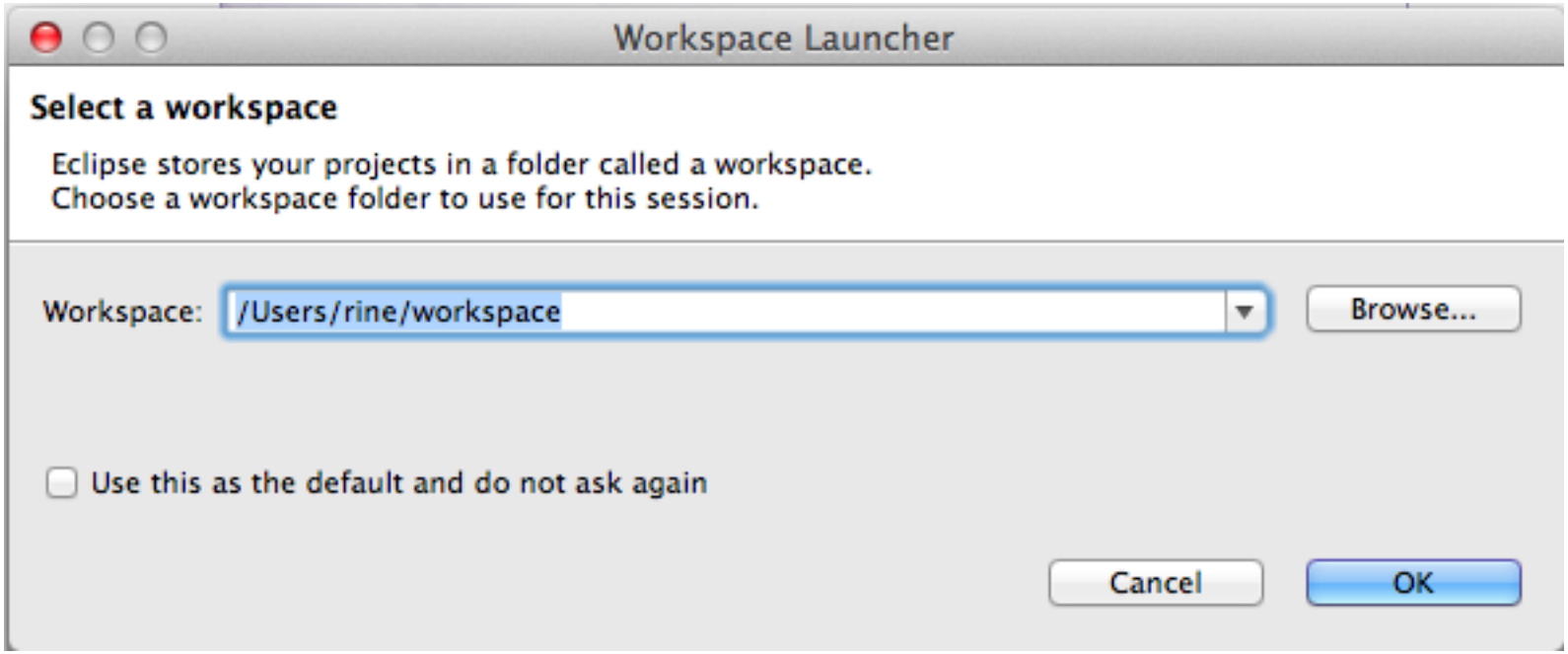
右クリック(Run As), (メニュ), メニュ

表示（コンソール）

Eclipseを使ってみよう

Eclipseの起動と終了

〇Eclipseのダブルクリック 



workspace が基本作業場

画像処理2　EclipseによるJava

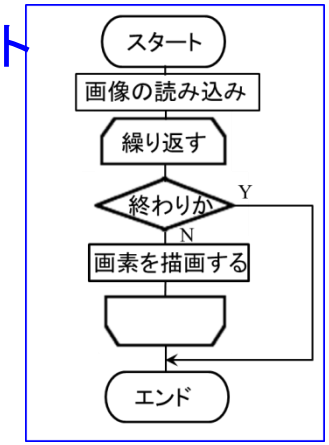
2022.12.10

プログラミングによる問題解決

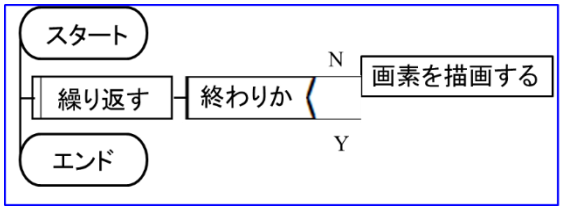
目的（問題解決）	アルゴリズム	言語プログラミング	変換する	実行

画像をコピーする

フローチャート



PAD



言語化

Java言語
C言語
Fortran
COBOL

自動化

その他

ビルド（コンパイル＋リンク）

実行

壁

PAD（Problem Analysis Diagram，日立，二村良彦，1979）

画像処理2　EclipseによるJava

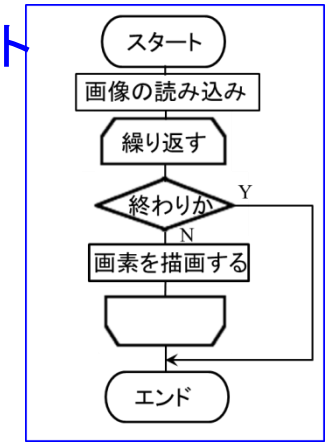
2022.12.03

プログラミングによる問題解決（Scratchまで）

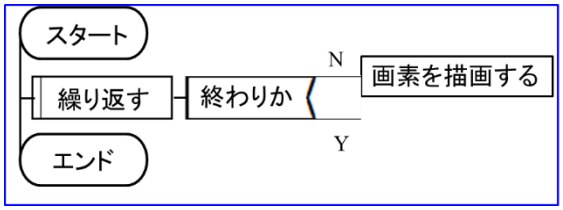
目的（問題解決）	アルゴリズム	言語プログラミング	変換する	実行

画像をコピーする

フローチャート



PAD



言語化

Java言語
C言語
Fortran
COBOL

その他

ビルド（コンパイル＋リンク）

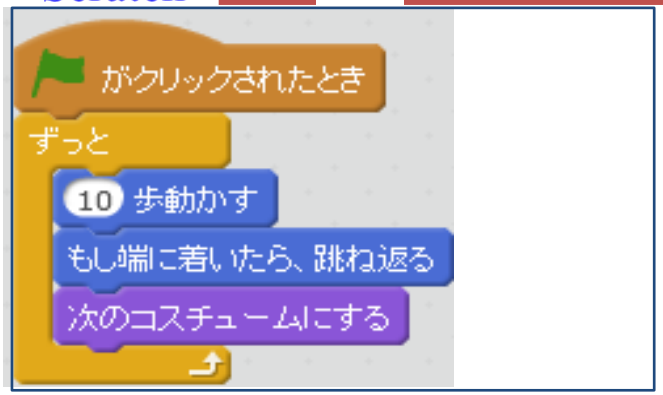
自動化

実行

Scratch

壁

PAD（Problem Analysis Diagram，日立，二村良彦，1979）



MITメディアラボ（2006）

はじめてのフレーム継承アプリケーション

```
package gazou;
import java.awt.*;//Frame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.event.*;                                    //イベント用，*：任意

public class Framecopi extends Frame {                      // Frame継承
        private static final long serialVersionUID = 1L;    //他の環境でも安全のため
                Image img;                                  //Image変数

        public Framecopi(String title) {
                setTitle(title);                            //タイトル設定
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {//閉じるボタン対応
                                System.exit(0);             //終了する
                        }
        });
        init();
        }

        public void init(){                                  //画像データの取得
                Toolkit tk;                                 //Toolkitメソッド
                tk = Toolkit.getDefaultToolkit();
                img = tk.getImage("image/test5.jpg");
                MediaTracker mt=new MediaTracker(this);     //画像が読み込み完了するまで監視
                mt.addImage(img,0);                         //メディアトラッカーする上での対象のデータと配列0
                try{                                        //例外を処理をする構文
                        mt.waitForID(0);                    //例外が発生しそうなプログラム
                } catch (InterruptedException e){}          //例外の場合の処理

        }

        public void paint(Graphics g) {
                g.drawImage(img, 10, 10, this);             //画像表示
        }

        public static void main(String[] args) {
                Framecopi frm = new Framecopi("画像表示");//「画像表示」タイトルのフレーム生成を変数frmで準備
                frm.setSize(800, 600);                      //窓サイズ横、縦
                frm.setVisible(true);                       //フレームを表示する
        }
}
```

はじめてのフレーム継承アプリケーション

```
package gazou;
import java.awt.*;//Frame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.event.*;                                    //イベント用，*：任意

public class Framecopi extends Frame {                      // Frame継承
        private static final long serialVersionUID = 1L;    //他の環境でも安全のため
                Image img;                                  //Image変数

        public Framecopi(String title) {
                setTitle(title);                            //タイトル設定
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {//閉じるボタン対応
                                System.exit(0);             //終了する
                        }
        });
        init();
        }

        public void init(){                                  //画像データの取得
                Toolkit tk;                                 //Toolkitメソッド
                tk = Toolkit.getDefaultToolkit();
                img = tk.getImage("image/test5.jpg");
                MediaTracker mt=new MediaTracker(this);     //画像が読み込み完了するまで監視
                mt.addImage(img,0);                         //メディアトラッカーする上での対象のデータと配列0
                try{                                        //例外を処理をする構文
                        mt.waitForID(0);                    //例外が発生しそうなプログラム
                } catch (InterruptedException e){}          //例外の場合の処理

        }

        public void paint(Graphics g) {
                g.drawImage(img, 10, 10, this);             //画像表示
        }

        public static void main(String[] args) {
                Framecopi frm = new Framecopi("画像表示");//「画像表示」タイトルのフレーム生成を変数frmで準備
                frm.setSize(800, 600);                      //窓サイズ横、縦
                frm.setVisible(true);                       //フレームを表示する
        }
}
```

作ったclassはどこ？

はじめてのフレーム継承アプリケーション

```
package gazou;
import java.awt.*;//Frame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.event.*;                    //イベント用，*：任意

public class Framecopi1 extends Frame {      // Frame継承
        private static final long serialVersionUID = 1L;//他の環境でも安全のため
                Image img;                          //Image変数

        public Framecopi1(String title) {
                setSize(800, 600);                  //窓サイズ横、縦
        setVisible(true);             //フレームを表示する
                setTitle(title);                    //タイトル設定
        addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {//閉じるボタン対応
                        System.exit(0);             //終了する
                }
        });
        init();
   }

        public void init(){
                                                            //画像データの取得
        Toolkit tk;                                     //Toolkitメソッド
        tk = Toolkit.getDefaultToolkit();
        img = tk.getImage("image/test1.jpg");
        MediaTracker mt=new MediaTracker(this);   //画像が読み込み完了するまで監視
                mt.addImage(img,0);                         //メディアトラッカーする上での対象のデータと配列0
                try{                                    //例外を処理をする構文
                        mt.waitForID(0);                    //例外が発生しそうなプログラム
                } catch (InterruptedException e){}      //例外の場合の処理
   }

   public void paint(Graphics g) {
   g.drawImage(img, 10, 10, this);              //画像表示
   }

        public static void main(String[] args) {
                new Framecopi1("画像表示");                 //「画像表示」タイトルのフレーム生成
 }
}
```

作ったclassはどこ？

画像のコピープログラム

参考プログラム・・・後記

・プログラムは理解してその論理を真似る
・アルゴリズム（処理手順）が大切

プログラムの説明

・変数
・配列，　画像取得
・イメージ処理
・画素の扱い，ビット処理，論理演算，表示

デバッグの方法ほか

文法エラー

・文法書ほか，本

論理エラー

処理が目的どおり動作しているか
・アルゴリズムの理解
・変数変化の理解と表示

画面描画のソースプログラム（参考）

```
package gazou;
import java.awt.*;//Frame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.event.*;                                    //イベント用，*：任意
import java.awt.image.PixelGrabber;

public class Framedisp extends Frame {                      // Frame継承
    private static final long serialVersionUID = 1L;        //他の環境でも安全のため

    int pix[];                                              //イメージの1ピクセルずつのデータを格納する配列
    int w=0,h=0,wh;
    PixelGrabber pixelG;                                    //を画像読込みのピクセルグラバーメソッドの宣言
    Image img;                                              //Image変数

    public static void main(String[] args) {
        Framedisp frm=new Framedisp("画像表示");            //「画像表示」タイトルのフレーム生成を変数frmで準備
        frm.setSize(1000, 800);                             //窓サイズ横、縦
        frm.setVisible(true);                               //フレームを表示する
    }

    public Framedisp(String title) {
        setTitle(title);                                    //タイトル設定
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {//閉じるボタン対応
                System.exit(0);                             //終了する
            }
        });
        init();
    }

    public void init(){                                     //画像データの取得
        Toolkit tk;                                         //Toolkitメソッド
        tk = Toolkit.getDefaultToolkit();
        img = tk.getImage("image/test1.jpg");
        MediaTracker mt=new MediaTracker(this);             //画像が読み込み完了するまで監視
        mt.addImage(img,0);                                 //メディアトラッカーする上での対象のデータと配列0
```

```
        try{                                                    //例外を処理をする構文
            mt.waitForID(0);                                    //例外が発生しそうなプログラム
        } catch (InterruptedException e){}                      //例外の場合の処理
        w=img.getWidth(this);                                   //読込み画像の横サイズ
        h=img.getHeight(this);                                  //読込み画像の縦サイズ
        System.out.println ("w="+w+"   h="+h);                  //コンソールへの表示
        wh=w*h;                                                 //whは画素数の最大値

        pix=new int[w*h];                                       //読み込み画像の配列pixの領域
        pixelG=new PixelGrabber(img,0,0,w,h,pix,0,w);//ピクセルグラバーメソッドの配列生成
                                                                //(抽出画像の幅,高さ,配列の番号,配列の幅)
        try{                                                    //例外処理をする構文
            pixelG.grabPixels();                                //例外が発生しそうなプログラム
        } catch(InterruptedException e) {}                      //例外処理の割り込み
    }

    public void paint(Graphics g) {
        int ix,iy,i;
        int red,green,blue;                                     //ピクセルの各色,
        int iw=1,ih=iw;                                         //表示ピクセルのサイズを決める
        g.drawImage(img, 10, 10, this);                         //画像表示
        for (i=0;i<wh;i++){
            ix=(int)(i%w);
            iy=(int)(i/w);
            if (i>wh){
                break;
            }
            red=((pix[i]>>16)&0xff);                            //r系統のカラーを設定する論理演算子
            green=((pix[i]>>8)&0xff);                           //g系統のカラーを設定する論理演算子
            blue=((pix[i]&0xff));                               //b系統のカラーを設定する論理演算子
            g.setColor(new Color(red,green,blue));              //カラー情報を各配列にset
            g.fillRect(ix+w+100,iy+10,iw,ih);                   //表示位置調整
        }
    }
}
```

ビットのシフト演算とビット演算

（例）`pix[i]>>16　//pix[i]の内容を16ビット右にシフトする`

演算子	使用例	意　味
<<	$a << 3$	a を 左へ3ビットシフト
>>	$a >> 3$	a を 右へ3ビットシフト(符号有り)
>>>	$a >>> 3$	a を 右へ3ビットシフト(符号無し)

（例）`((pix[i]>>16)&0xff);//シフト演算結果と0xffの論理積`

演算子	使用例	意　味
~	!A	NOT（ビットの否定）
&	A & B	AND（ビットごとの論理積）
^	A ^ B	XOR（ビットごとの排他的論理和）
|	A | B	OR（ビットごとの論理和）

練習・デバッグほか

- 変数名を変更する
  int, double
- 読込み画像を変更する
  *.jpg, *.png, ほか
- 描画位置, http://www.ipc.hokusei.ac.jp/~z00103/Edu/Java/Applet/appletGraphics_01.html
  x-, y- 座標
- フレーム（Window)の大きさ
- デバッグ
  ix, iy計算の%と/が反対の場合
- ビットシフト数を変える

【課題】: 画像描画の方法　　(画像はtest5.jpgとする）

1）表示枠を拡大し，一次元配列pix[]の内容を+600pixel離れた右隣に表示する。
2）ix, iy計算の%と/を反対にして表示し，異なる理由を調べる。
3）ビットシフト演算のシフト数8や16を変えると表示色が変わる。何故か？

Applet継承の場合（参考）

```
package gazou;
import java.applet.Applet;
import java.awt.*;
import java.awt.image.*;
import java.awt.Graphics;

public class Copipe extends Applet{
        private static final long serialVersionUID=1L;//安全策，このクラスはシリアライズ可能
        int pix[];//イメージの1ピクセルずつのデータを格納する配列
        int w,h,wh;

        PixelGrabber pixelG;//を画像読込みのピクセルグラバーメソッドの宣言
        Image img;  //読み込み画像配列のimage宣言

        public void init(){  //アプレットの初期化
                img=getImage(getCodeBase(), "image/test5.jpg"); //ディレクトリから画像を読み込む
                MediaTracker mt=new MediaTracker(this);//読み込み中に勝手に出力などをさせないメソッド
                mt.addImage(img,0);  // メディアトラッカーする上での対象のデータと配列0
                try{                    //例外を処理をする構文
                        mt.waitForID(0); //例外が発生しそうなプログラム
                } catch (InterruptedException e){}    //例外処理の割り込み

                w=img.getWidth(this);//読込み画像の横サイズ
                h=img.getHeight(this);//読込み画像の縦サイズ
                System.out.println ("w="+w+"   h="+h);
                wh=w*h;
                pix=new int[wh];    //読み込み画像の配列pixの領域
                pixelG=new PixelGrabber(img,0,0,w,h,pix,0,w);
                //ピクセルグラバーメソッドの配列生成
                //（抽出画像の幅,高さ,配列の番号,配列の幅）
```

```
        try{          //例外処理をする構文
            pixelG.grabPixels();   //例外が発生しそうなプログラム
        } catch(InterruptedException e) {}  //例外処理の割り込み
    }

    public void paint(Graphics g){
        int ix,iy,i;
        int iw=1; //表示ピクセルのサイズを決める
        int ih=iw;
        int red,green,blue;//ピクセルの各色, whは画素数の最大値

        g.drawImage(img, 10, 10, this);    //画像表示

        for (i=0;i<wh;i++){
            ix=(int)(i%w);
            iy=(int)(i/w);
            if (i>wh){
                break;
            }
            red=((pix[i]>>16)&0xff);  //r系統のカラーを設定する論理演算子
            green=((pix[i]>>8)&0xff); //g系統のカラーを設定する論理演算子
            blue=((pix[i]&0xff));     //b系統のカラーを設定する論理演算子
            g.setColor(new Color(red,green,blue)); //カラー情報を各配列にset
            g.fillRect(ix+w,iy+10,iw,ih);
        }
    }
}
```

はじめてのフレーム継承アプリケーション


```
package gazou;
import java.awt.*;//Frame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.event.*;                                    //イベント用，*：任意

public class Framecopi extends Frame {                      // Frame継承
        private static final long serialVersionUID = 1L;    //他の環境でも安全のため
                Image img;                                  //Image変数

        public Framecopi(String title) {
                setTitle(title);                            //タイトル設定
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {//閉じるボタン対応
                                System.exit(0);             //終了する
                        }
        });
        init();
        }

        public void init(){                                  //画像データの取得
                Toolkit tk;                                 //Toolkitメソッド
                tk = Toolkit.getDefaultToolkit();
                img = tk.getImage("image/test5.jpg");
                MediaTracker mt=new MediaTracker(this);     //画像が読み込み完了するまで監視
                mt.addImage(img,0);                         //メディアトラッカーする上での対象のデータと配列0
                try{                                        //例外を処理をする構文
                        mt.waitForID(0);                    //例外が発生しそうなプログラム
                } catch (InterruptedException e){}          //例外の場合の処理

        }

        public void paint(Graphics g) {
                g.drawImage(img, 10, 10, this);             //画像表示
        }

        public static void main(String[] args) {
                Framecopi frm = new Framecopi("画像表示");//「画像表示」タイトルのフレーム生成を変数frmで準備
                frm.setSize(800, 600);                      //窓サイズ横、縦
                frm.setVisible(true);                       //フレームを表示する
        }
}
```

はじめてのフレーム継承アプリケーション

```
package gazou;
import java.awt.*;//Frame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.event.*;                                    //イベント用，*：任意

public class Framecopi extends Frame {                      // Frame継承
        private static final long serialVersionUID = 1L;    //他の環境でも安全のため
                Image img;                                  //Image変数

        public Framecopi(String title) {
                setTitle(title);                            //タイトル設定
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {//閉じるボタン対応
                                System.exit(0);             //終了する
                        }
        });
        init();
        }

        public void init(){                                  //画像データの取得
                Toolkit tk;                                 //Toolkitメソッド
                tk = Toolkit.getDefaultToolkit();
                img = tk.getImage("image/test5.jpg");
                MediaTracker mt=new MediaTracker(this);     //画像が読み込み完了するまで監視
                mt.addImage(img,0);                         //メディアトラッカーする上での対象のデータと配列0
                try{                                        //例外を処理をする構文
                        mt.waitForID(0);                    //例外が発生しそうなプログラム
                } catch (InterruptedException e){}          //例外の場合の処理

        }

        public void paint(Graphics g) {
                g.drawImage(img, 10, 10, this);             //画像表示
        }

        public static void main(String[] args) {
                Framecopi frm = new Framecopi("画像表示");//「画像表示」タイトルのフレーム生成を変数frmで準備
                frm.setSize(800, 600);                      //窓サイズ横、縦
                frm.setVisible(true);                       //フレームを表示する
        }
}
```

作ったclassはどこ？

はじめてのフレーム継承アプリケーション

```
package gazou;
import java.awt.*;//Frame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.event.*;                    //イベント用，*：任意

public class Framecopi1 extends Frame {      // Frame継承
        private static final long serialVersionUID = 1L;//他の環境でも安全のため
                Image img;                          //Image変数

        public Framecopi1(String title) {
                setSize(800, 600);                  //窓サイズ横、縦
        setVisible(true);             //フレームを表示する
                setTitle(title);                    //タイトル設定
        addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {//閉じるボタン対応
                        System.exit(0);             //終了する
                }
        });
        init();
   }

        public void init(){
                                                            //画像データの取得
        Toolkit tk;                                     //Toolkitメソッド
        tk = Toolkit.getDefaultToolkit();
        img = tk.getImage("image/test1.jpg");
        MediaTracker mt=new MediaTracker(this);   //画像が読み込み完了するまで監視
                mt.addImage(img,0);                         //メディアトラッカーする上での対象のデータと配列0
                try{                                    //例外を処理をする構文
                        mt.waitForID(0);                    //例外が発生しそうなプログラム
                } catch (InterruptedException e){}      //例外の場合の処理
   }

   public void paint(Graphics g) {
   g.drawImage(img, 10, 10, this);              //画像表示
   }

        public static void main(String[] args) {
                new Framecopi1("画像表示");                 //「画像表示」タイトルのフレーム生成
 }
}
```

作ったclassはどこ？

画像のコピープログラム

参考プログラム・・・後記

・プログラムは理解してその論理を真似る
・アルゴリズム（処理手順）が大切

プログラムの説明

・変数
・配列，　画像取得
・イメージ処理
・画素の扱い，ビット処理，論理演算，表示

デバッグの方法ほか

文法エラー

・文法書ほか，本

論理エラー

処理が目的どおり動作しているか
・アルゴリズムの理解
・変数変化の理解と表示

画面描画のソースプログラム（参考）

```
package gazou;
import java.awt.*;//Frame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.event.*;                                    //イベント用，*：任意
import java.awt.image.PixelGrabber;

public class Framedisp extends Frame {                      // Frame継承
    private static final long serialVersionUID = 1L;        //他の環境でも安全のため

    int pix[];                                              //イメージの1ピクセルずつのデータを格納する配列
    int w=0,h=0,wh;
    PixelGrabber pixelG;                                    //を画像読込みのピクセルグラバーメソッドの宣言
    Image img;                                              //Image変数

    public static void main(String[] args) {
        Framedisp frm=new Framedisp("画像表示");            //「画像表示」タイトルのフレーム生成を変数frmで準備
        frm.setSize(1000, 800);                             //窓サイズ横、縦
        frm.setVisible(true);                               //フレームを表示する
    }

    public Framedisp(String title) {
        setTitle(title);                                    //タイトル設定
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {//閉じるボタン対応
                System.exit(0);                             //終了する
            }
        });
        init();
    }

    public void init(){                                     //画像データの取得
        Toolkit tk;                                         //Toolkitメソッド
        tk = Toolkit.getDefaultToolkit();
        img = tk.getImage("image/test1.jpg");
        MediaTracker mt=new MediaTracker(this);             //画像が読み込み完了するまで監視
        mt.addImage(img,0);                                 //メディアトラッカーする上での対象のデータと配列0
```

```
        try{                                                    //例外を処理をする構文
            mt.waitForID(0);                                    //例外が発生しそうなプログラム
        } catch (InterruptedException e){}                      //例外の場合の処理
        w=img.getWidth(this);                                   //読込み画像の横サイズ
        h=img.getHeight(this);                                  //読込み画像の縦サイズ
        System.out.println ("w="+w+"   h="+h);                  //コンソールへの表示
        wh=w*h;                                                 //whは画素数の最大値

        pix=new int[w*h];                                       //読み込み画像の配列pixの領域
        pixelG=new PixelGrabber(img,0,0,w,h,pix,0,w);//ピクセルグラバーメソッドの配列生成
                                                                //(抽出画像の幅,高さ,配列の番号,配列の幅)
        try{                                                    //例外処理をする構文
            pixelG.grabPixels();                                //例外が発生しそうなプログラム
        } catch(InterruptedException e) {}                      //例外処理の割り込み
    }

    public void paint(Graphics g) {
        int ix,iy,i;
        int red,green,blue;                                     //ピクセルの各色,
        int iw=1,ih=iw;                                         //表示ピクセルのサイズを決める
        g.drawImage(img, 10, 10, this);                         //画像表示
        for (i=0;i<wh;i++){
            ix=(int)(i%w);
            iy=(int)(i/w);
            if (i>wh){
                break;
            }
            red=((pix[i]>>16)&0xff);                            //r系統のカラーを設定する論理演算子
            green=((pix[i]>>8)&0xff);                           //g系統のカラーを設定する論理演算子
            blue=((pix[i]&0xff));                               //b系統のカラーを設定する論理演算子
            g.setColor(new Color(red,green,blue));              //カラー情報を各配列にset
            g.fillRect(ix+w+100,iy+10,iw,ih);                   //表示位置調整
        }
    }
}
```

ビットのシフト演算とビット演算

（例）`pix[i]>>16　//pix[i]の内容を16ビット右にシフトする`

演算子	使用例	意　味
<<	$a << 3$	a を 左へ3ビットシフト
>>	$a >> 3$	a を 右へ3ビットシフト(符号有り)
>>>	$a >>> 3$	a を 右へ3ビットシフト(符号無し)

（例）`((pix[i]>>16)&0xff);//シフト演算結果と0xffの論理積`

演算子	使用例	意　味
~	!A	NOT（ビットの否定）
&	A & B	AND（ビットごとの論理積）
^	A ^ B	XOR（ビットごとの排他的論理和）
|	A | B	OR（ビットごとの論理和）

練習・デバッグほか

- 変数名を変更する
  int, double
- 読込み画像を変更する
  *.jpg, *.png, ほか
- 描画位置, http://www.ipc.hokusei.ac.jp/~z00103/Edu/Java/Applet/appletGraphics_01.html
  x-, y- 座標
- フレーム（Window)の大きさ
- デバッグ
  ix, iy計算の%と/が反対の場合
- ビットシフト数を変える

【課題】: 画像描画の方法　　(画像はtest5.jpgとする）

1）表示枠を拡大し，一次元配列pix[]の内容を+600pixel離れた右隣に表示する。
2）ix, iy計算の%と/を反対にして表示し，異なる理由を調べる。
3）ビットシフト演算のシフト数8や16を変えると表示色が変わる。何故か？

Applet継承の場合（参考）

```
package gazou;
import java.applet.Applet;
import java.awt.*;
import java.awt.image.*;
import java.awt.Graphics;

public class Copipe extends Applet{
        private static final long serialVersionUID=1L;//安全策，このクラスはシリアライズ可能
        int pix[];//イメージの1ピクセルずつのデータを格納する配列
        int w,h,wh;

        PixelGrabber pixelG;//を画像読込みのピクセルグラバーメソッドの宣言
        Image img;  //読み込み画像配列のimage宣言

        public void init(){  //アプレットの初期化
                img=getImage(getCodeBase(), "image/test5.jpg"); //ディレクトリから画像を読み込む
                MediaTracker mt=new MediaTracker(this);//読み込み中に勝手に出力などをさせないメソッド
                mt.addImage(img,0);  // メディアトラッカーする上での対象のデータと配列0
                try{                    //例外を処理をする構文
                        mt.waitForID(0); //例外が発生しそうなプログラム
                } catch (InterruptedException e){}    //例外処理の割り込み

                w=img.getWidth(this);//読込み画像の横サイズ
                h=img.getHeight(this);//読込み画像の縦サイズ
                System.out.println ("w="+w+"   h="+h);
                wh=w*h;
                pix=new int[wh];    //読み込み画像の配列pixの領域
                pixelG=new PixelGrabber(img,0,0,w,h,pix,0,w);
                //ピクセルグラバーメソッドの配列生成
                //（抽出画像の幅,高さ,配列の番号,配列の幅）
```

```
        try{          //例外処理をする構文
            pixelG.grabPixels();   //例外が発生しそうなプログラム
        } catch(InterruptedException e) {}  //例外処理の割り込み
    }

    public void paint(Graphics g){
        int ix,iy,i;
        int iw=1; //表示ピクセルのサイズを決める
        int ih=iw;
        int red,green,blue;//ピクセルの各色, whは画素数の最大値

        g.drawImage(img, 10, 10, this);    //画像表示

        for (i=0;i<wh;i++){
            ix=(int)(i%w);
            iy=(int)(i/w);
            if (i>wh){
                break;
            }
            red=((pix[i]>>16)&0xff);  //r系統のカラーを設定する論理演算子
            green=((pix[i]>>8)&0xff); //g系統のカラーを設定する論理演算子
            blue=((pix[i]&0xff));     //b系統のカラーを設定する論理演算子
            g.setColor(new Color(red,green,blue)); //カラー情報を各配列にset
            g.fillRect(ix+w,iy+10,iw,ih);
        }
    }
}
```

GUI(Graphical User Interface)の利用

2022.12.22

AWT（Abstract Windowing Tools）からSwing
（GUIアプリケーションを作成するためのクラスライブラリ）

AWT

```
import java.awt.Color;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Toolkit;
import java.awt.event.WindowAdapter; //イベント用，*：任意
import java.awt.event.WindowEvent;
import java.awt.image.PixelGrabber;
```

Swing

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JButton;
import java.awt.event.*;
```

```
Image img;          //Imageオブジェクト
```

AWTの調べ方

2022.12.22

https://docs.oracle.com/javase/jp/8/docs/api/java/awt/package-summary.html

パッケージ java.awt

ユーザー・インタフェースの作成およびグラフィックスとイメージのペイント用のすべてのクラスを含みます。

参照: 説明

インタフェースのサマリー

インタフェース	説明
ActiveEvent	自分自身をディスパッチできるイベントのためのインタフェースです。
Adjustable	ある制限範囲内に含まれる調整可能な数値を持つオブジェクト用のインタフェースです。
Composite	Compositeインタフェースは、CompositeContextとともに、基本となるグラフィックス領域に描画プリミティブを構成するためのメソッドを定義します。
CompositeContext	CompositeContextインタフェースは、合成操作のためにカプセル化され、最適化された環境を定義します。
ItemSelectable	項目の集まりを持つオブジェクトに対するインタフェースです。ゼロまたはそれ以上の項目を選択することができます。
KeyEventDispatcher	KeyEventDispatcherは、すべてのKeyEventsのターゲット指定とディスパッチに関して現在のKeyboardFocusManagerと協力します。

Swingの調べ方

2022.12.22

https://docs.oracle.com/javase/jp/8/docs/api/javax/swing/package-summary.html

概要 パッケージ クラス 使用 階層ツリー 非推奨 索引 ヘルプ Java(t Standa

前のパッケージ 次のパッケージ フレーム フレームなし すべてのクラス

パッケージ javax.swing

すべてのプラットフォーム上で可能なかぎり同じように機能する「軽量」(Java共通言語)コンポーネントのセットを提供しま

参照: 説明

インタフェースのサマリー

インタフェース	説明
Action	Actionインタフェースは、同じ機能が複数のコントロールによってアクセスされる場 ActionListenerインタフェースに対する便利な拡張機能を提供します。
BoundedRangeModel	SliderやProgressBarなどのコンポーネントが使用するデータ・モデルを定義します
ButtonModel	ボタンの状態モデルです。
CellEditor	すべての汎用エディタが実装可能なメソッドを定義します。
ComboBoxEditor	JComboBoxコンポーネントに使われるエディタ・コンポーネントです。
ComboBoxModel<E>	コンボボックスのデータ・モデルです。

画像処理4 セピア色

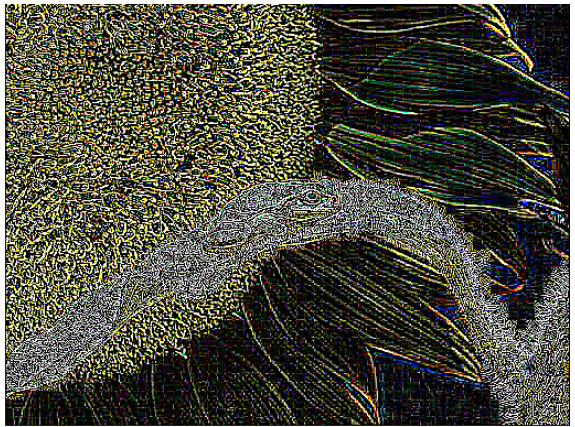
2022.12.22

元画像　　フィルター処理の結果

赤色フィルター

セピアフィルター

微分フィルター



画像処理4 セピア色

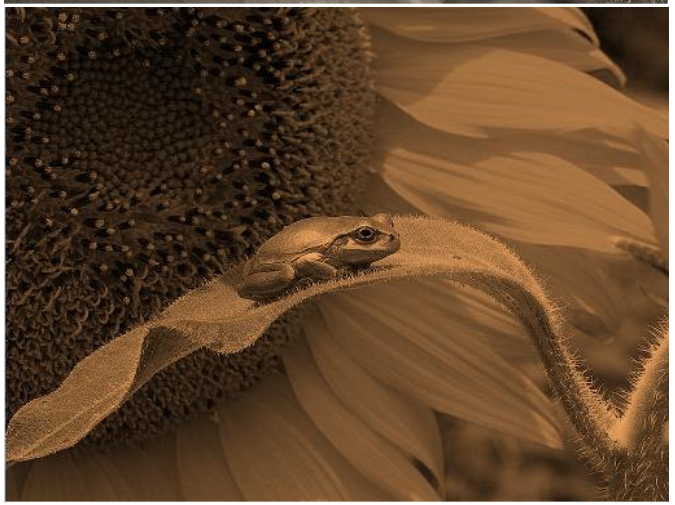
1）セピア色の重みが（R，G，B）＝（107，74，43）の場合

カラー情報(R, G, B)=(red, green, blue)のmean=(red+green+blue)/3 を使って（R，G，B）＝(mean*107/107, mean*74/107, mean*43/107)などの重みをつけて表示する。

・重み（R，G，B）＝(160, 132, 96)など

セピア1

セピア2



```
public void paint(Graphics g) {

      int cnt=0,ix,iy,n,mean,iw=1,ih=iw;
      g.drawImage(img, 0, 0, this);//元画像の表示
      cnt++;
      System.out.println ("w="+w+"  h="+h+"  cnt="+cnt);
      for (n=0;n<wh;n++){
            ix=(int)(n%w);
            iy=(int)(n/w);
            if (n>wh){
                  break;
            }
            int red=(pix[n]>>16)&0xff;
            int green=(pix[n]>>8)&0xff;
            int blue=(pix[n]&0xff);
            mean=(red+green+blue)/3;
            //g.setColor(new Color(mean,mean*110/130,mean*90/130));//画像表示
            //g.setColor(new Color(mean,mean*74/107,mean*43/107));//画像表示
            //g.setColor(new Color(mean,mean,mean));//画像表示
            g.setColor(new Color(red,green,blue));//画像表示
            g.fillRect(ix+550,iy,iw,ih);
        }
}
```

変数と浮動小数点数の扱い

変数	char	2Byte（16bit）	Unicode文字 ¥u0000～¥uFFFF
	byte	1Byte（8bit）	整数 -128～127
	short	2Byte（16bit）	整数 -32768～32767
	int	4Byte（32bit）	整数 -2147483648～2147483647
	long	8Byte（64bit）	整数 -9223372036854775808～9223372036854775807
	float	4Byte（32bit）	単精度浮動小数点数
	double	8Byte（64bit）	倍精度浮動小数点数
	string		文字列，オブジェクト

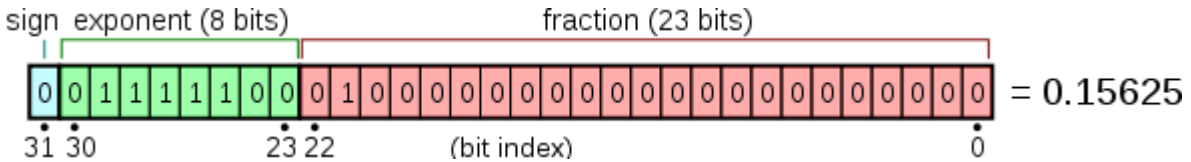
数の表記

整数　n bit　$-2^{n-1} \sim 2^{n-1}-1$

浮動小数点

32bit

IEEE 754 単精度



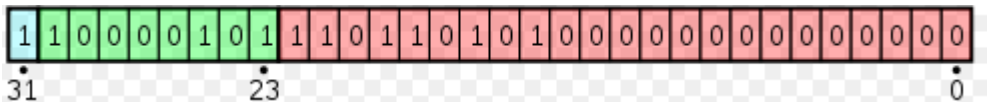
$(-118.625)_{10}$の表現，　負数なので，符号bitは 1 となる。

$|-118.625| = (118.\ 625)_{10} = (1110110.101)_2 = (1.110110101)_2 \times 2^6$

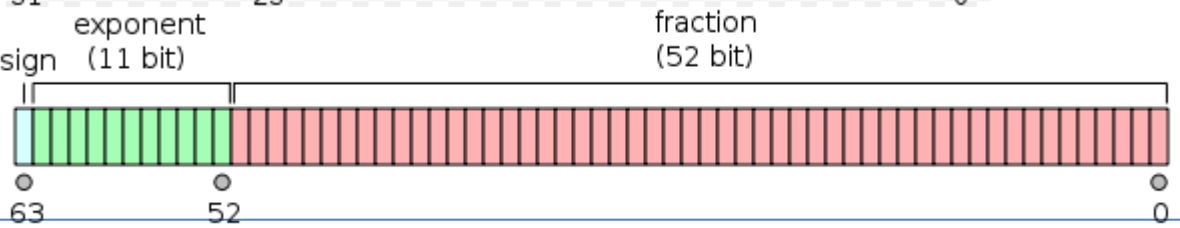
Fraction部分は1を除いた1未満の部分で$(11011010100000000000000)_2$ である。

指数の6は6+127（バイアス）＝133＝$(10000101)_2$である。

32bit



64bit



キャスト(変数型変換)プログラム

変数			
	int	4Byte(32bit)	整数 -2147483648～2147483647
	long	8Byte(64bit)	整数 -9223372036854775808～9223372036854775807
	float	4Byte(32bit)	単精度浮動小数点数
	double	8Byte(64bit)	倍精度浮動小数点数

参考プログラム

画像処理5 カラーフィルター

2023.01.07

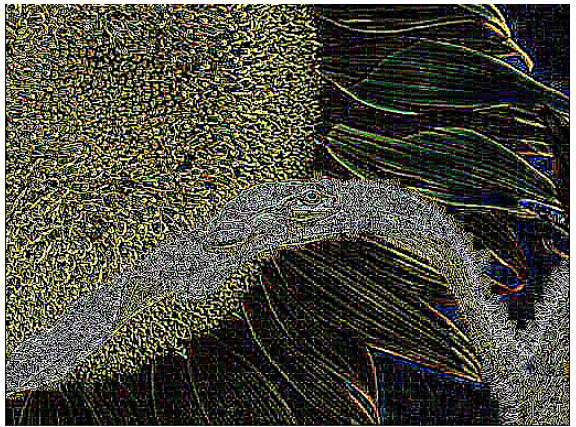
元画像

フィルター処理

赤色フィルター

セピアフィルター

微分フィルター



Red フィルター

```
package gshori;

//JFrame用GUI（グラフィカルユーザインターフェース）ライブラリ，awt:abstract windowing tools）
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Toolkit;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.image.PixelGrabber;

import javax.swing.JFrame;

public class JFilter extends JFrame {                    // JFrame継承

        int pix[];                                       //イメージの1ピクセルずつのデータを格納する配列
        int w=0,h=0,wh;
        PixelGrabber pixelG;                             //を画像読込みのピクセルグラバーメソッドの宣言
        Image img;                                       //Image変数

        public static void main(String[] args) {
                JFilter frm=new JFilter("画像表示");//「画像表示」タイトルのフレーム生成を変数frmで準備
                frm.setSize(1300, 800);                  //窓サイズ横、縦
                frm.setVisible(true);                    //フレームを表示する
         }

        public JFilter(String title) {
                setTitle(title);                         //タイトル設定
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {//閉じるボタン対応
                                System.exit(0);          //終了する
                        }
                });
                init();
         }

        public void init(){                              //画像データの取得
                Toolkit tk;                              //Toolkitメソッド
                tk = Toolkit.getDefaultToolkit();
                img = tk.getImage("images/test5.jpg");
                MediaTracker mt=new MediaTracker(this);   //画像が読み込み完了するまで監視
                mt.addImage(img,0);
```

```
        try{                                            //例外を処理をする構文
            mt.waitForID(0);                            //例外が発生しそうなプログラム
        } catch (InterruptedException e){}              //例外の場合の処理
        w=img.getWidth(this);                           //読込み画像の横サイズ
        h=img.getHeight(this);                          //読込み画像の縦サイズ
        System.out.println ("w="+w+"   h="+h);
        wh=w*h;                                         //whは画素数の最大値

        pix=new int[wh];                                //読み込み画像の配列pixの領域
        pixelG=new PixelGrabber(img,0,0,w,h,pix,0,w);//ピクセルグラバーメソッドの配列生成
                                                        //（抽出画像の幅,高さ,配列の番号,配列の幅）
        try{                                            //例外処理をする構文
            pixelG.grabPixels();                        //例外が発生しそうなプログラム
        } catch(InterruptedException e) {}              //例外処理の割り込み
    }

    public void paint(Graphics g) {
        int ix,iy,i;
        int red,green=0,blue=0;                         //ピクセルの各色,
        int iw=1,ih=iw;                                 //表示ピクセルのサイズを決める
        g.drawImage(img, 0, 0, this);                   //画像表示
        //Red_filter
        for (i=0;i<wh;i++){
            ix=(int)(i%w);
            iy=(int)(i/w);
            if (i>wh){
                break;
            }
            red=((pix[i]>>16)&0xff);                    //r系統のカラーを設定する論理演算子
            g.setColor(new Color(red,green,blue));//色設定
            g.fillRect(ix+w+50,iy,iw,ih);
        }
    }
}
```

グレイフィルター

グレイにするとは

カラー情報 (R, G, B)=(0, 0, 0)は黒，(R, G, B)=(255, 255, 255)は白
その中間の数値aの(R, G, B)=(a, a, a)は灰色
　だから
元画像の各pixelのカラー情報(R, G, B)=(red, green, blue)の平均値
mean=(red+green+blue)/3 を使った(R, G, B)=(mean, mean, mean)は
そのpixelのカラー情報の平均的明るさでのグレイ色になる。

練習:グレイフィルター(表示部のみ)

```
public void paint(Graphics g) {
    int ix,iy,i;
    int red,green,blue,mean;//ピクセルの各色,平均値
    int iw=1,ih=iw;    //表示ピクセルのサイズを決める
    g.drawImage(img, 0, 0, this);    //画像表示
    //グレイ_filter
    for (i=0;i<wh;i++){
        ix=(int)(i%w);
        iy=(int)(i/w);
        if (i>wh){
            break;
        }
        red=((pix[i]>>16)&0xff);//r系統のカラーを設定する論理演算子
        green=((pix[i]>>8)&0xff);//g系統のカラーを設定する論理演算子
        blue=((pix[i])&0xff);//b系統のカラーを設定する論理演算子
        mean=(red+green+blue)/3;
        g.setColor(new Color(mean,mean,mean));//平均値の色設定
        g.fillRect(ix+w+50,iy,iw,ih);
    }
}
```

練習：フィルター・クラスによるREDフィルター

```
package gshori;

import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Toolkit;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.image.FilteredImageSource;
import java.awt.image.ImageFilter;
import java.awt.image.PixelGrabber;
import javax.swing.JFrame;

public class FilterC extends JFrame {// Frame継承

	int pix[];//イメージの1ピクセルずつのデータを格納する配列
	int w=0,h=0,wh;
	PixelGrabber pixelG;//を画像読込みのピクセルグラバーメソッドの宣言
	Image img;//Image変数
	Image new_img;

	public static void main(String[] args) {
		FilterC frm=new FilterC("画像表示");
		//「画像表示」タイトルのフレーム生成を変数frmで準備
		frm.setSize(1300, 800);		//窓サイズ横、縦
		frm.setVisible(true);	//フレームを表示する
	}

	public FilterC(String title) {
		setTitle(title);//タイトル設定
		 addWindowListener(new WindowAdapter() {
				public void windowClosing(WindowEvent e) {//閉じるボタン対応
					System.exit(0);		//終了する
				}
		});
		init();
	}
```

```
    public void init(){       //画像データの取得
        Toolkit tk;//Toolkitメソッド
        tk = Toolkit.getDefaultToolkit();
        img = tk.getImage("images/test5.jpg");
        MediaTracker mt=new MediaTracker(this);//画像が読み込み完了するまで監視
        mt.addImage(img,0);
        try{                //例外を処理をする構文
                    mt.waitForID(0); //例外が発生しそうなプログラム
        } catch (InterruptedException e){} //例外の場合の処理
        w=img.getWidth(this);//読込み画像の横サイズ
        h=img.getHeight(this);//読込み画像の縦サイズ
        System.out.println ("w="+w+"   h="+h);
        wh=w*h;//whは画素数の最大値

        pix=new int[wh];   //読み込み画像の配列pixの領域
        pixelG=new PixelGrabber(img,0,0,w,h,pix,0,w);//ピクセルグラバーメソッドの配列生成
        //（抽出画像の幅,高さ,配列の番号,配列の幅）
        Try{          //例外処理をする構文
                    pixelG.grabPixels();   //例外が発生しそうなプログラム
        } catch(InterruptedException e) {} //例外処理の割り込み
        filter_shori();
    }

    void filter_shori(){
        ImageFilter f= new Red_Filter(); //フィルタークラスの呼び出し

        FilteredImageSource fi = new FilteredImageSource(img.getSource(), f);
        new_img = createImage(fi);
    }

    public void paint(Graphics g) {
        g.drawImage(img, 0, 0, this);    //画像表示
        g.drawImage(new_img, w+50, 0, this);
    }
}
```

Redフィルター

```
package gshori;
//Red_Filter.java
//
import java.awt.*;
import java.awt.image.*;

class Red_Filter extends RGBImageFilter {  //java.awtパッケージの中のRGBImageFilterクラス

        public Red_Filter(){    //コンストラクタ：クラスと同一名の特別な自動実行メソッド
                canFilterIndexColorModel=true;
                        //すべてのPixelに対して同じ処理をする場合，RGBImageFilterサブクラス
                        //のフィルター処理されるIndexColorModelの変数をtrueに設定する
                        //特定のPixelについての処理はfalseとする
        }

        public int filterRGB(int x, int y, int rgb ){
                        // filterRGBメソッドはRGB値を受け取りメソッド内の処理を行う
                        // 1 つの入力ピクセルを 1 つの出力ピクセルに変換する
                return rgb&0xffff0000;// Red値だけを残し，G(reen),B(lue)値を0にする
        }
}
```

二つのプログラムをリンクする
コンパイル順に注意

他の色フィルター

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

class Red_Filter extends RGBImageFilter { //java.awtパッケージの中のRGBImageFilterクラス

        public Red_Filter(){
                canFilterIndexColorModel = true;
                  //RGBImageFilterクラスをtrueに設定するは「ピクセルの位置に関係なく
                  //すべてのピクセルにたいして処理する
                  //特定のピクセルについての処理はfalseとする
        }

        public int filterRGB(int x, int y, int rgb ){
                        // filterRGBメソッドはRGB値を受け取りメソッド内の処理を行う
                       // 1 つの入力ピクセルを 1 つの出力ピクセルに変換する
                return rgb&0xffff0000;// ←色フィルターの設定（Red)
                //return rgb&0xff00ff00;// ←色フィルターの設定（Green)
        }
}
```

filterRGB(int x, int y, int rgb)メソッドについて

https://docs.oracle.com/javase/jp/6/api/java/awt/image/RGBImageFilter.html

java.awt.image
クラス RGBImageFilter
このクラスは、デフォルト RGB ColorModel イメージのピクセルを修正する ImageFilter を容易に作成するための方法を提供します。

```
public int filterRGB(int x, int y, int rgb ){
 // filterRGBメソッドはRGB値を受け取りメソッド内の処理を行う
 // 1 つの入力ピクセルを 1 つの出力ピクセルに変換する
        return rgb&0xffff0000;// R値だけを残し，G,B値を 0 にする
```

パラメータ:
x - ピクセルの X 座標
y - ピクセルの Y 座標
rgb - デフォルト RGB カラーモデルの整数型ピクセル表現 戻り値:
デフォルト RGB カラーモデルのフィルタ処理されたピクセル

説明：java.awt.imageパッケージ

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

class Red_Filter extends RGBImageFilter {
                //java.awtパッケージの中のRGBImageFilterクラスを継承するRed_Filter

        public Red_Filter(){
                canFilterIndexColorModel = true;
        }

        public int filterRGB(int x, int y, int rgb ){// rgbにピクセル情報がある
                return rgb&0xffff0000;// 他の色に設定する
        }
}
```

//パッケージ内容は
https://docs.oracle.com/javase/jp/6/api/java/awt/image/package-summary.html
にある。そのクラスは全42
AffineTransformOp , AreaAveragingScaleFilter, BandCombineOp, BandedSampleModel,
BufferedImage, BufferedImageFilter, BufferStrategy, ByteLookupTable, ColorConvertOp, ColorModel, ComponentColorModel, ComponentSampleModel, ConvolveOp, CropImageFilter, DataBuffer, DataBufferByte, DataBufferDouble, DataBufferFloat,
ほか24

その中のフィルター機能

AreaAveragingScaleFilter

最近接点アルゴリズムよりもなめらかな結果が得られる，簡単な領域平均化アルゴリズムを使用してイメージをスケーリングする ImageFilter クラスです。

BufferedImageFilter

BufferedImageFilter クラスは、ImageFilter をサブクラス化し，転送元と転送先が単一のイメージ演算子 (BufferedImageOp) を使用して，Image Producer/Consumer/Observer パラダイムの BufferedImage にフィルタをかける簡易な手段を提供します。

CropImageFilter

イメージを切り出すための ImageFilter クラスです。

ReplicateScaleFilter

ImageFilter クラスは，もっとも簡単なアルゴリズムを使用してイメージのサイズを変更するクラスです。

RGBImageFilter

このクラスは，デフォルト RGB ColorModel イメージのピクセルを修正するなどがあり，利用できます。

最初にRGBImageFilterを使い，わかるようになってからその他のフィルターを使うようにしよう。

画像半分をセピア色にする（表示部分）

```
public void paint(Graphics g) {
        int ix,iy,i;
        int red,green,blue;//ピクセルの各色,
        int mean,iw=1,ih=iw;    //表示ピクセルのサイズを決める
        g.drawImage(img, 0, 0, this);   //画像表示
         //Red_filter
        for (i=0;i<wh;i++){
           ix=(int)(i%w);
           iy=(int)(i/w);
           if (i>wh){
           break;
           }
            red=(pix[i]>>16)&0xff;
           green=(pix[i]>>8)&0xff;
           blue=(pix[i]&0xff);
           mean=(red+green+blue)/3;
            if (ix<w/2) {
              g.setColor(new Color(red,green,blue));//色設定
              }else{
              g.setColor(new Color(mean,mean*110/130,mean*90/130));//セピア設定
              }
              g.fillRect(ix+w+50,iy,iw,ih);
           }
        }
}
```

課題：斜めセピア色にする

変数と浮動小数点数の扱い

変数	char	2Byte（16bit）	Unicode文字 ¥u0000～¥uFFFF
	byte	1Byte（8bit）	整数 -128～127
	short	2Byte（16bit）	整数 -32768～32767
	int	4Byte（32bit）	整数 -2147483648～2147483647
	long	8Byte（64bit）	整数 -9223372036854775808～9223372036854775807
	float	4Byte（32bit）	単精度浮動小数点数
	double	8Byte（64bit）	倍精度浮動小数点数
	string		文字列，オブジェクト

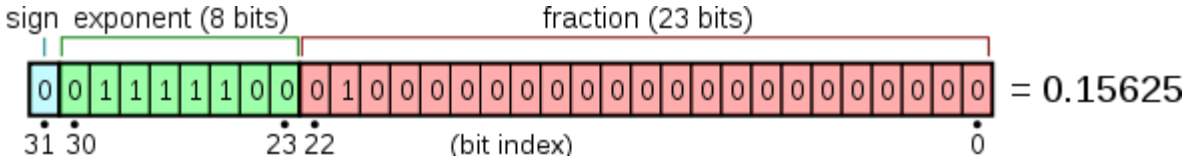
数の表記

整数　n bit　$-2^{n-1} \sim 2^{n-1}-1$

浮動小数点

32bit

IEEE 754 単精度



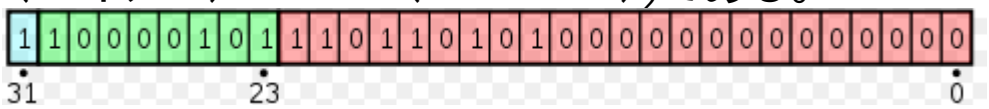
$(-118.625)_{10}$の表現，　負数なので，符号bitは 1 となる。

$|-118.625| = (118.\ 625)_{10} = (1110110.101)_2 = (1.110110101)_2 \times 2^6$

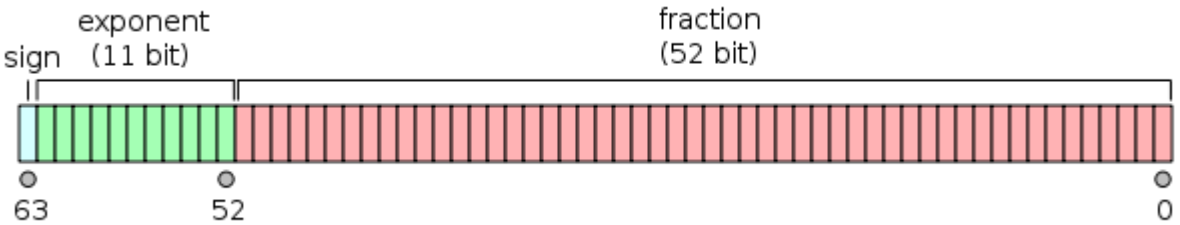
Fraction部分は1を除いた1未満の部分で$(11011010100000000000000)_2$ である。

指数の6は6+127（バイアス）＝133＝$(10000101)_2$である。

32bit



64bit



キャスト(変数型変換)プログラム

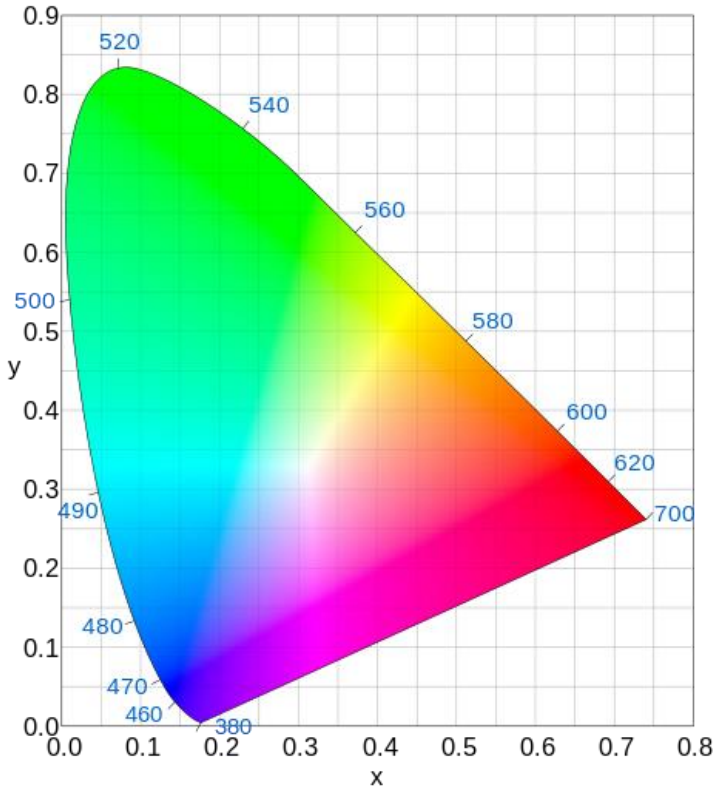
変数	int	4Byte(32bit)	整数 -2147483648～2147483647
	long	8Byte(64bit)	整数 -9223372036854775808～9223372036854775807
	float	4Byte(32bit)	単精度浮動小数点数
	double	8Byte(64bit)	倍精度浮動小数点数

参考プログラム

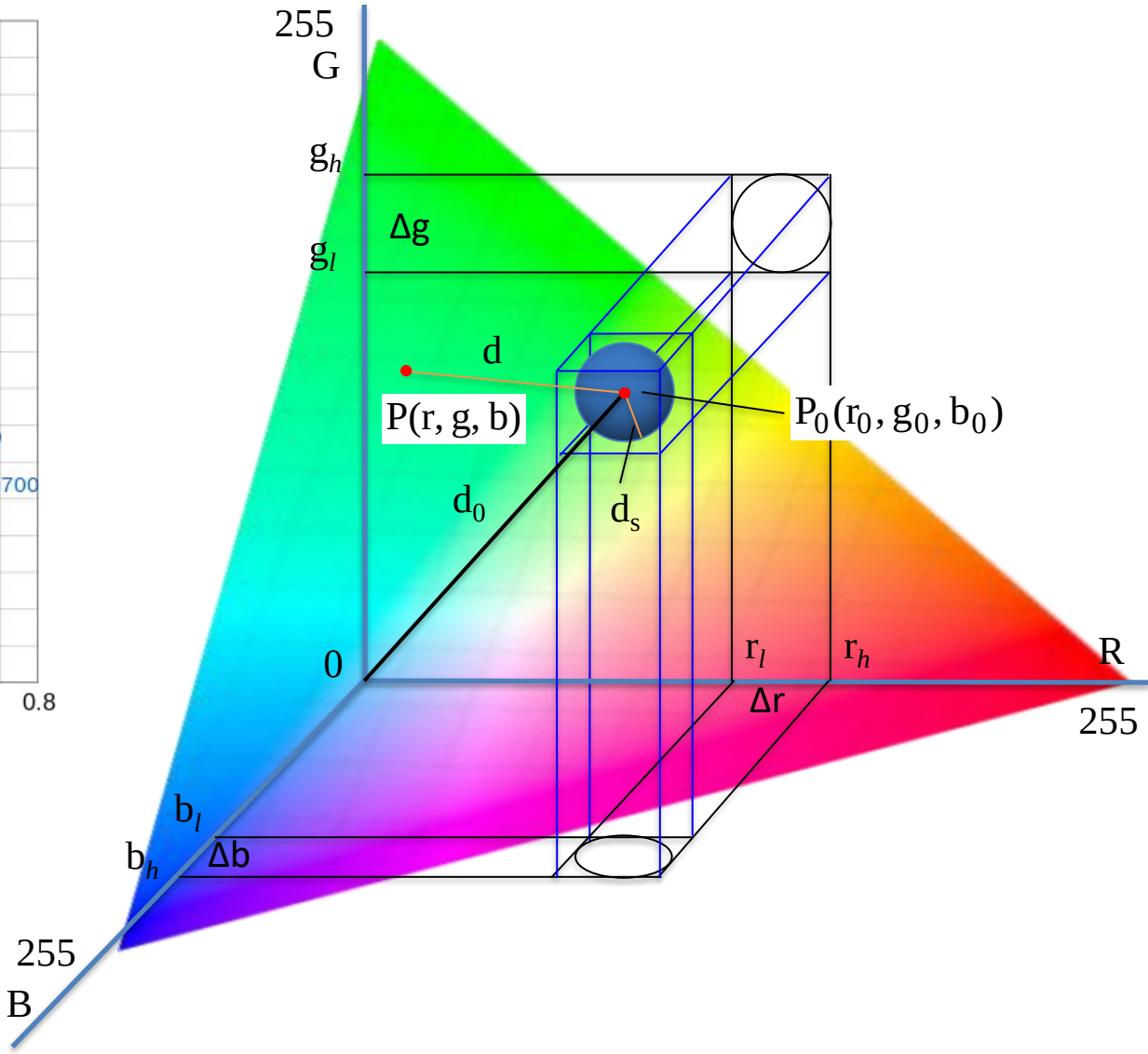
画像処理6　クロマキー(Chroma key)入門

2023.01.12

色空間での距離と範囲



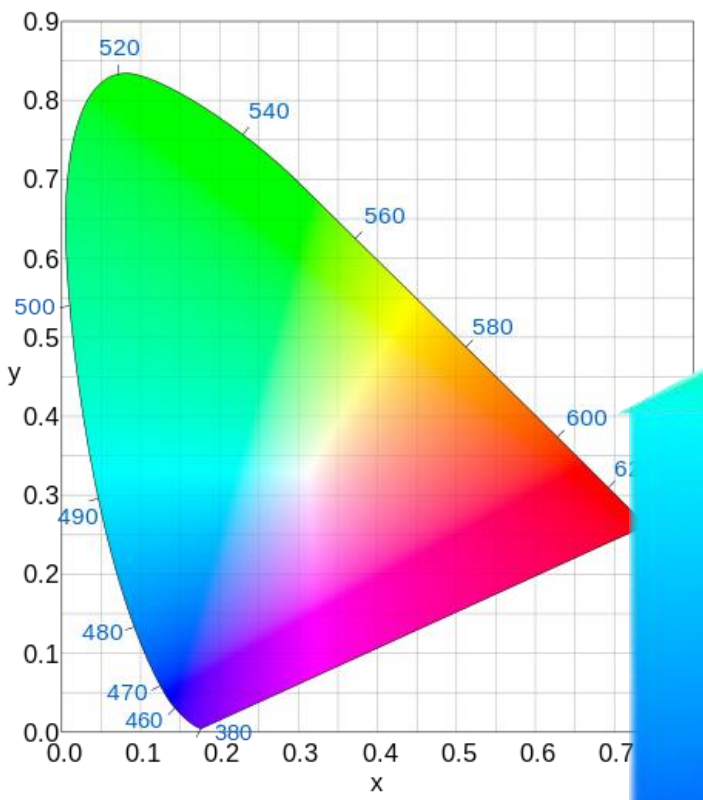
国際照明委員会（Commission internationale de l'éclairage，略称：CIE）の定める表色空間。

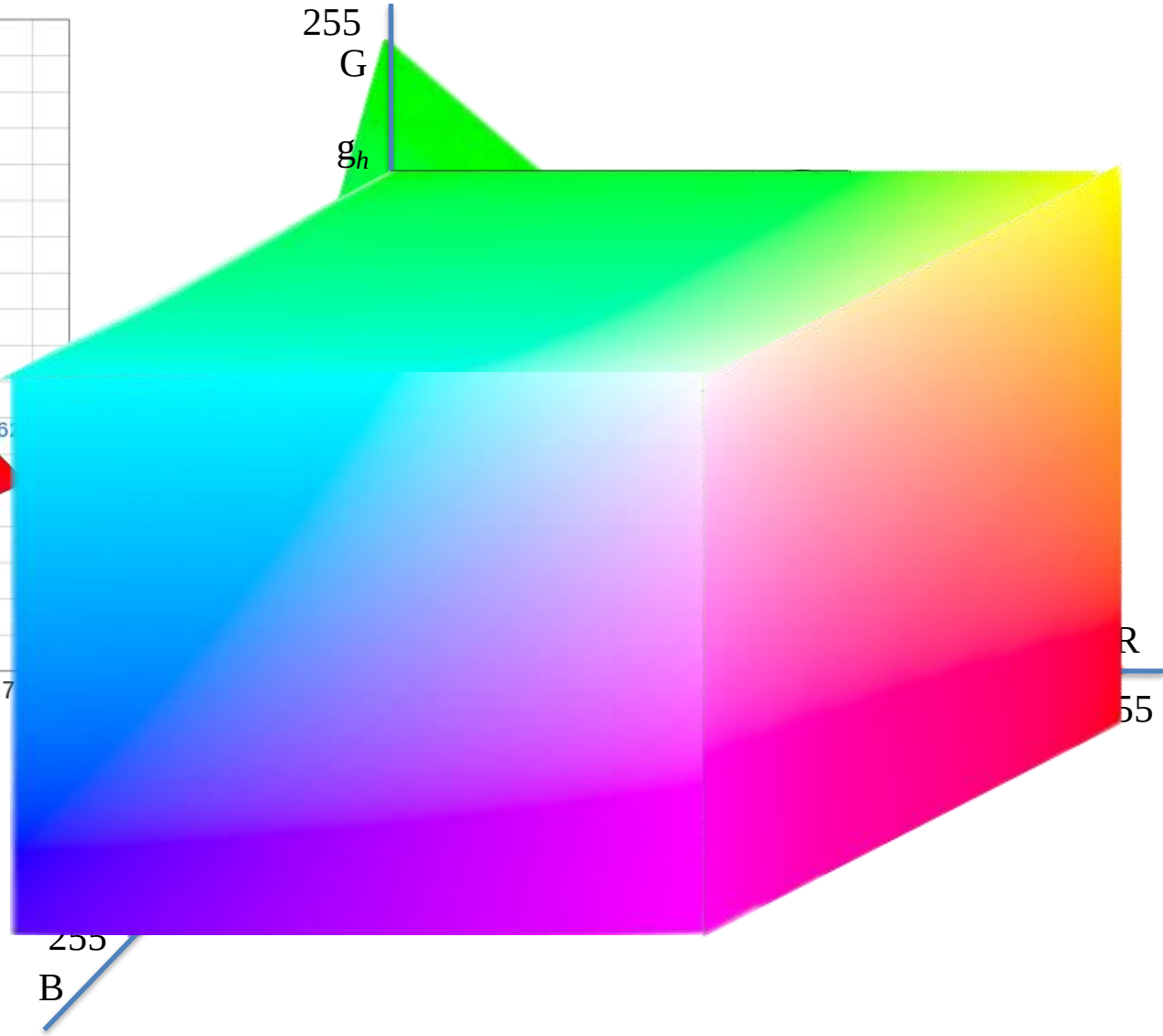


画像処理6　クロマキー(Chroma key)入門

2023.01.12

色空間での距離と範囲



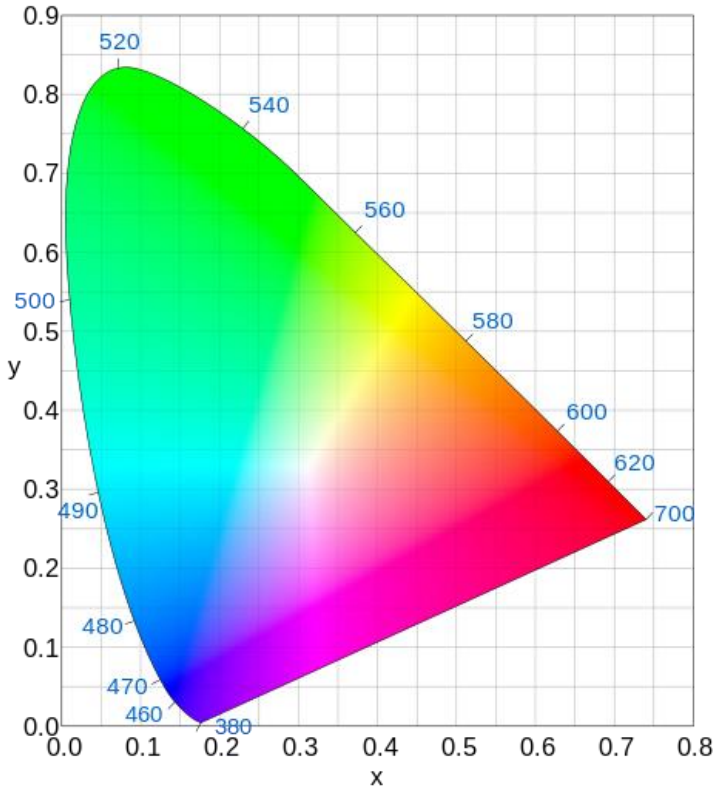


国際照明委員会（Commission internationale de l'éclairage，略称：CIE）の定める表色空間。

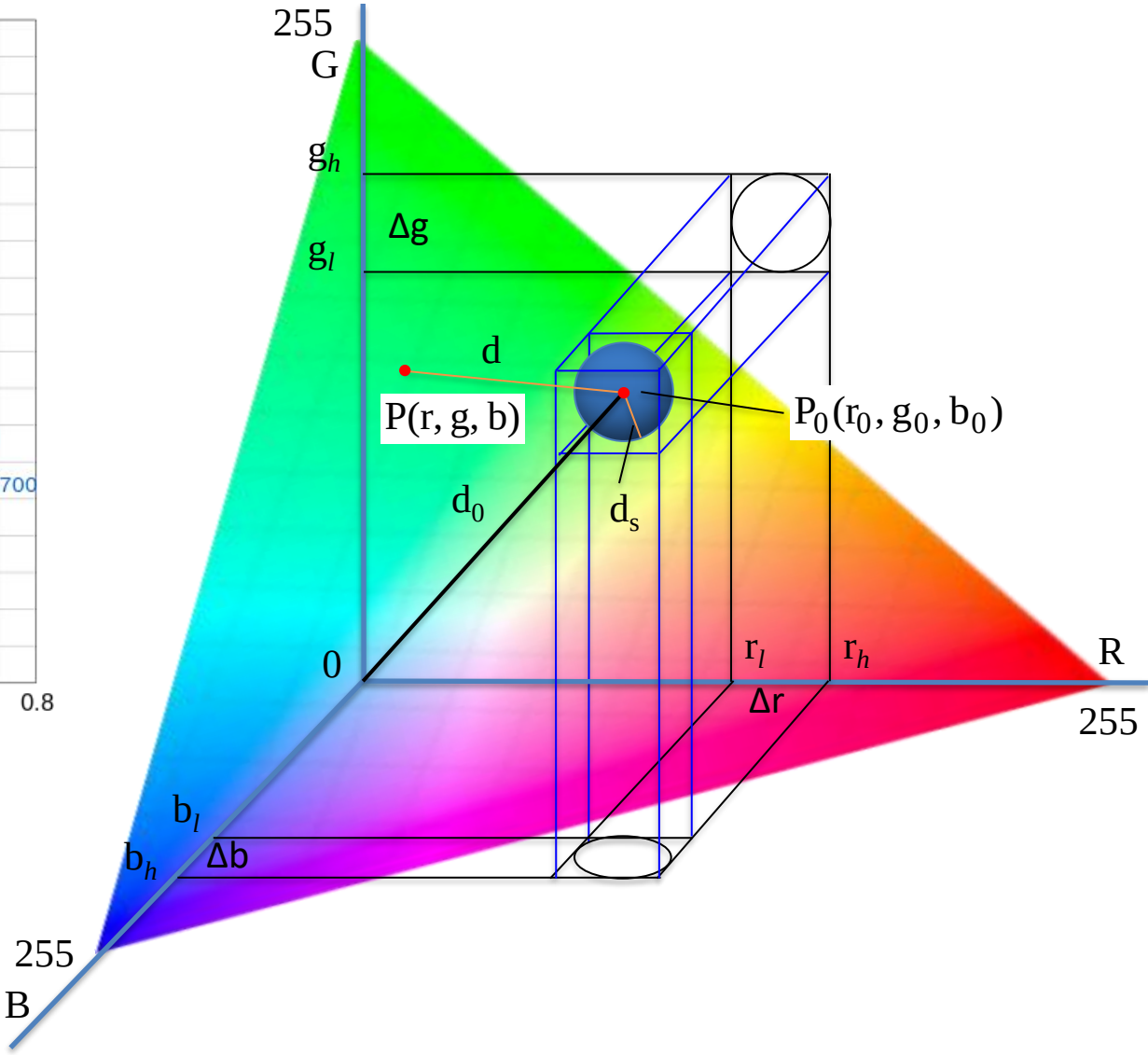
画像処理6　クロマキー(Chroma key)入門

2023.01.12

色空間での距離と範囲



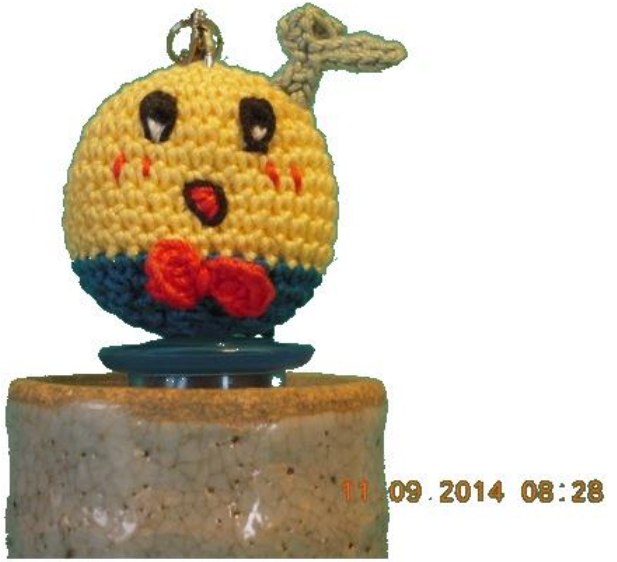
国際照明委員会（Commission internationale de l'éclairage，略称：CIE）の定める表色空間。



クロマキー処理の例

blue doll(test15.jpg)の場合

green doll(test16.jpg)の場合



クロマキー（色抜き）の概念

色距離(Color distance) d

図の記号　　Java言語

$\Delta r = r_h - r_l, \Rightarrow$ delr = redh – redl

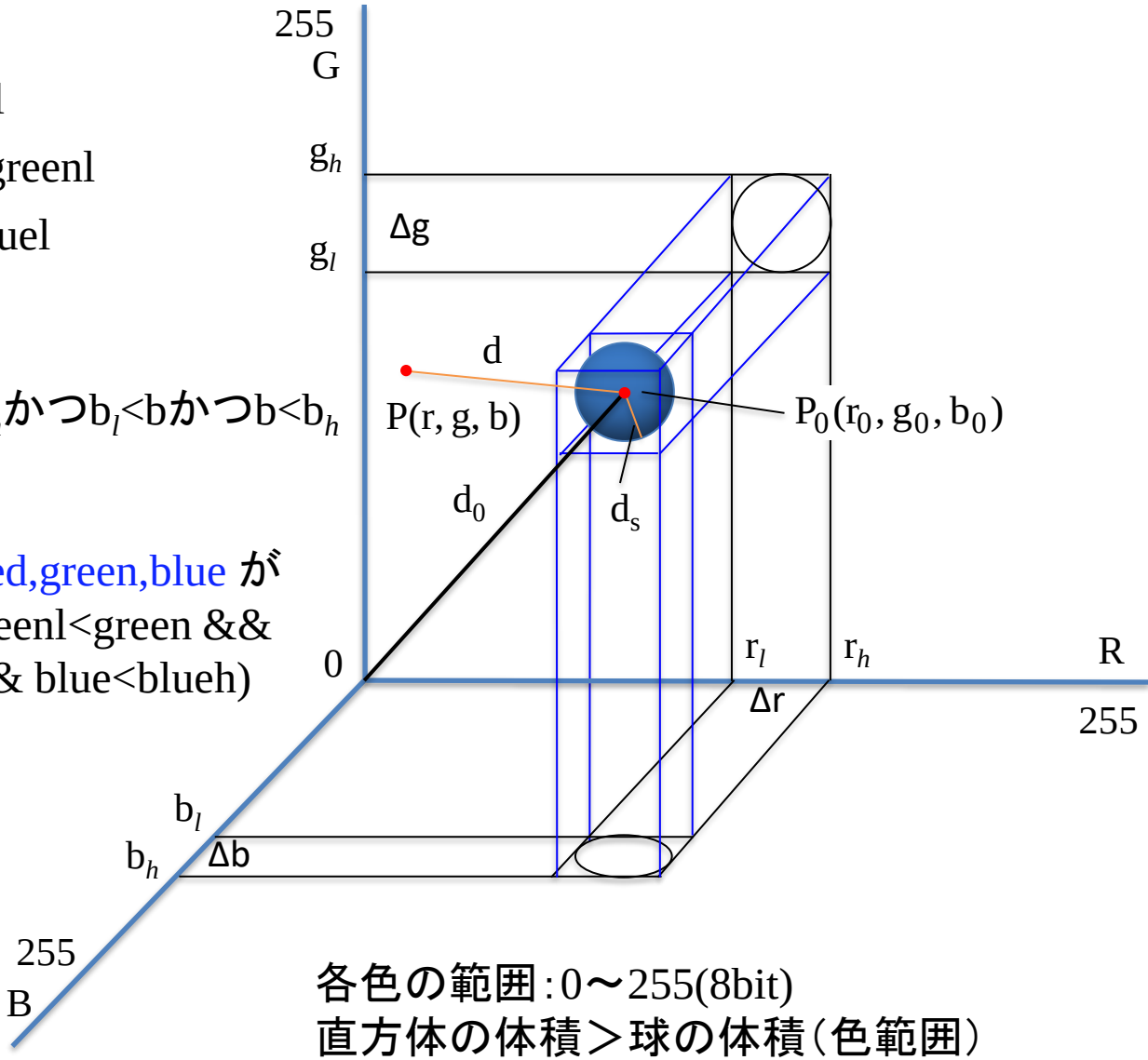
$\Delta g = g_h - g_l, \Rightarrow$ delg = greenh – greenl

$\Delta b = b_h - b_l, \Rightarrow$ delb = blueh – bluel

カラー直方体内の条件

r_l<rかつr<r_hかつg_l<gかつg<g_hかつb_l<bかつb<b_h

言語的には
各pixelについてのRGB情報　red,green,blue が
(redl<red && red<redh && greenl<green && green<greenh && bluel<blue && blue<blueh)
を満たしたとき表示しない。

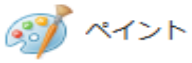


各色の範囲：0～255(8bit)
直方体の体積＞球の体積（色範囲）

クロマキー（色抜き）の簡単な方法

カラー直方体内の範囲を色抜きする

色情報をペイントなどで得る



アルゴリズムに合ったプログラムを書く
画像がtest21.jpg の場合

変数設定部分（例）

```
int redl=37,redh=88,greenl=50,greenh=97,bluel=95,blueh=152;//blue doll
```

表示部分

```
int red=(pix[n]>>16)&0xff;
int green=(pix[n]>>8)&0xff;
int blue=(pix[n]&0xff);
if (redl<red && red<redh && greenl<green && green<greenh && bluel<blue && blue<blueh){
        //カラー直方体の内部（色範囲）では何もしない，色を表示しない（色抜きする）
}else{//カラー直方体以外の色はそのまま表示する
            g.setColor(new Color(red,green,blue));
            g.fillRect(ix,iy,iw,ih);
}
```

コスモス画像の背景を色抜きしてみよう

blue flower(test17.jpg)

green flower(test18.jpg)

自然画像の一部を色抜きする

natural flower(test6.jpg)

コスモスの花のカラー直方体内の範囲（例）

int redl=126,redh=236,greenl=4,greenh=50,bluel=65,blueh=161;//cosmos flower

自然画像の一部を他の色に変更する

表示部分の参考プログラム（書き方）

```
public void paint(Graphics g) {
	int ix,iy,i;
	int redl=37,redh=88,greenl=50,greenh=97,bluel=95,blueh=140;//blue dollの場合
	int red,green,blue,mean;                    //ピクセルの各色,平均値
	int iw=1,ih=iw;                             //表示ピクセルのサイズを決める
	g.drawImage(img, 0, 0, this);               //画像表示
	for (i=0;i<wh;i++){
		ix=(int)(i%w);
		iy=(int)(i/w);
		if (i>wh){
			break;
		}
		red=(pix[i]>>16)&0xff;
		green=(pix[i]>>8)&0xff;
		blue=(pix[i]&0xff);
		if (redl<red && red<redh && greenl<green && green<greenh && bluel<blue &&
blue<blueh){                                    //pixelのRGB情報がｶﾗｰ直方体内の場合
                                                //なにも表示しない
		}else{
			g.setColor(new Color(red,green,blue));
			g.fillRect(ix+w+50,iy,iw,ih);
		}
	}
}
```

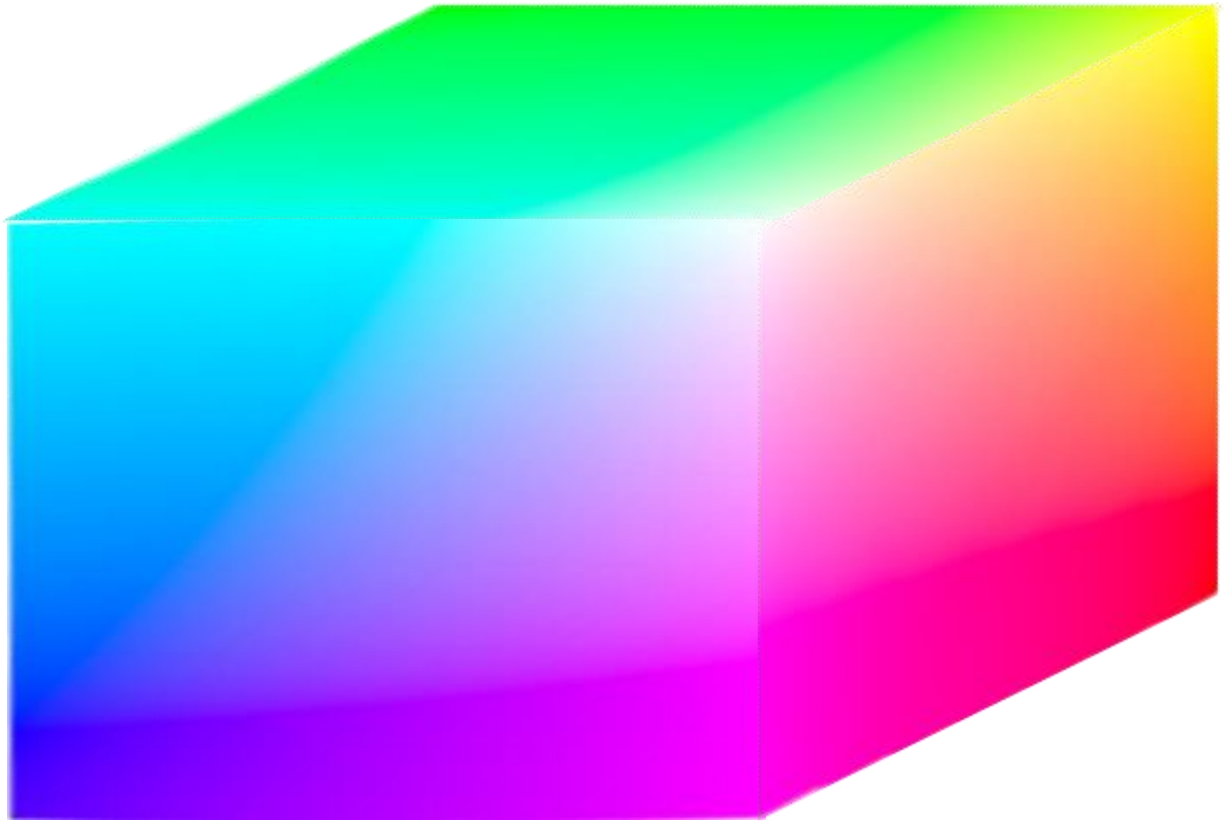
分かりやすく書く

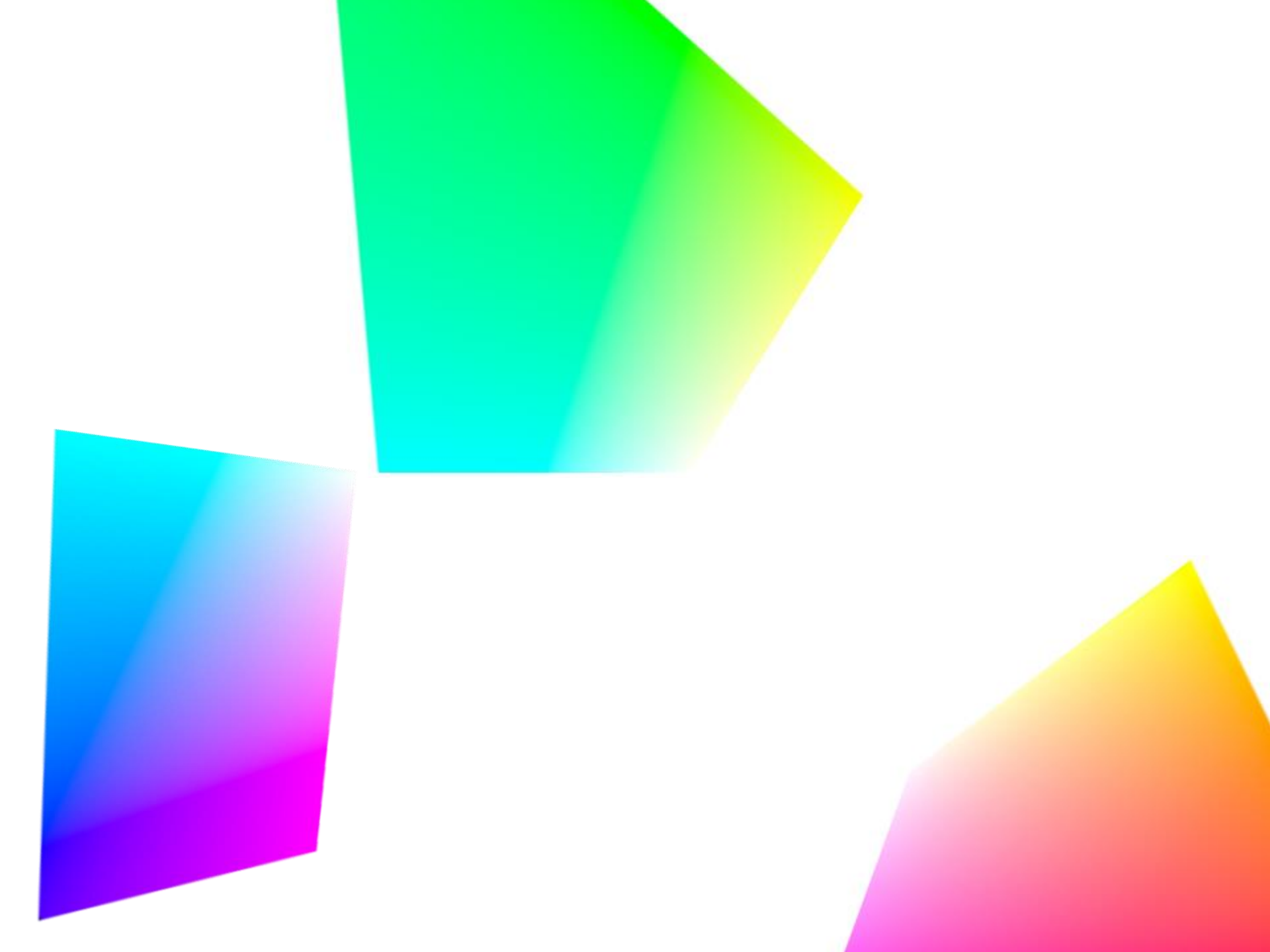
他の例

```
//int redl=45,redh=85,greenl=105,greenh=153,bluel=70,blueh=122;//green doll
//int redl=24,redh=71,greenl=51,greenh=104,bluel=104,blueh=185;//blue flower
//int redl=42,redh=72,greenl=121,greenh=154,bluel=126,blueh=154;//green flower
//int redl=108,redh=251,greenl=3,greenh=90,bluel=60,blueh=196;//natural flower
```

表示速度を上げるための参考プログラム

```
//                                      省略
Image img;//Imageオブジェクト
BufferedImage img2=null;                //追加
//                                      省略
                } catch(InterruptedException e) {}  //例外処理の割り込み
                filter_shori();         //追加
        }
        void filter_shori(){
                int i,red,green,blue;
                int redl=37,redh=88,greenl=50,greenh=97,bluel=95,blueh=140;//blue dollの場合
                for (i=0; i<wh; i++){
                        if (i>wh){
                                        break;
                        }
                        red=(pix[i]>>16)&0xff;
                        green=(pix[i]>>8)&0xff;
                        blue=(pix[i]&0xff);
                        if (redl<red && red<redh && greenl<green && green<greenh && bluel<blue && blue<
                                        pix[i]=0xffffffff;//ｶﾗｰ直方体内にて白抜き
                        }else{   ;
                        }
                }
                img2 = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);//幅w高さhのBufferedImageの
                img2.setRGB(0, 0, w, h, pix, 0, w);//開始点(0,  0)から幅w高さhあるピクセル配列pixを幅wでス
                // JPEG 形式で保存.
        }
        public void paint(Graphics g) {
                //画像表示
                g.drawImage(img, 0, 20, this);
                g.drawImage(img2,w+50,20,null);
        }
}
```





画像処理7　クロマキー(Chroma key)応用　2023.1.19

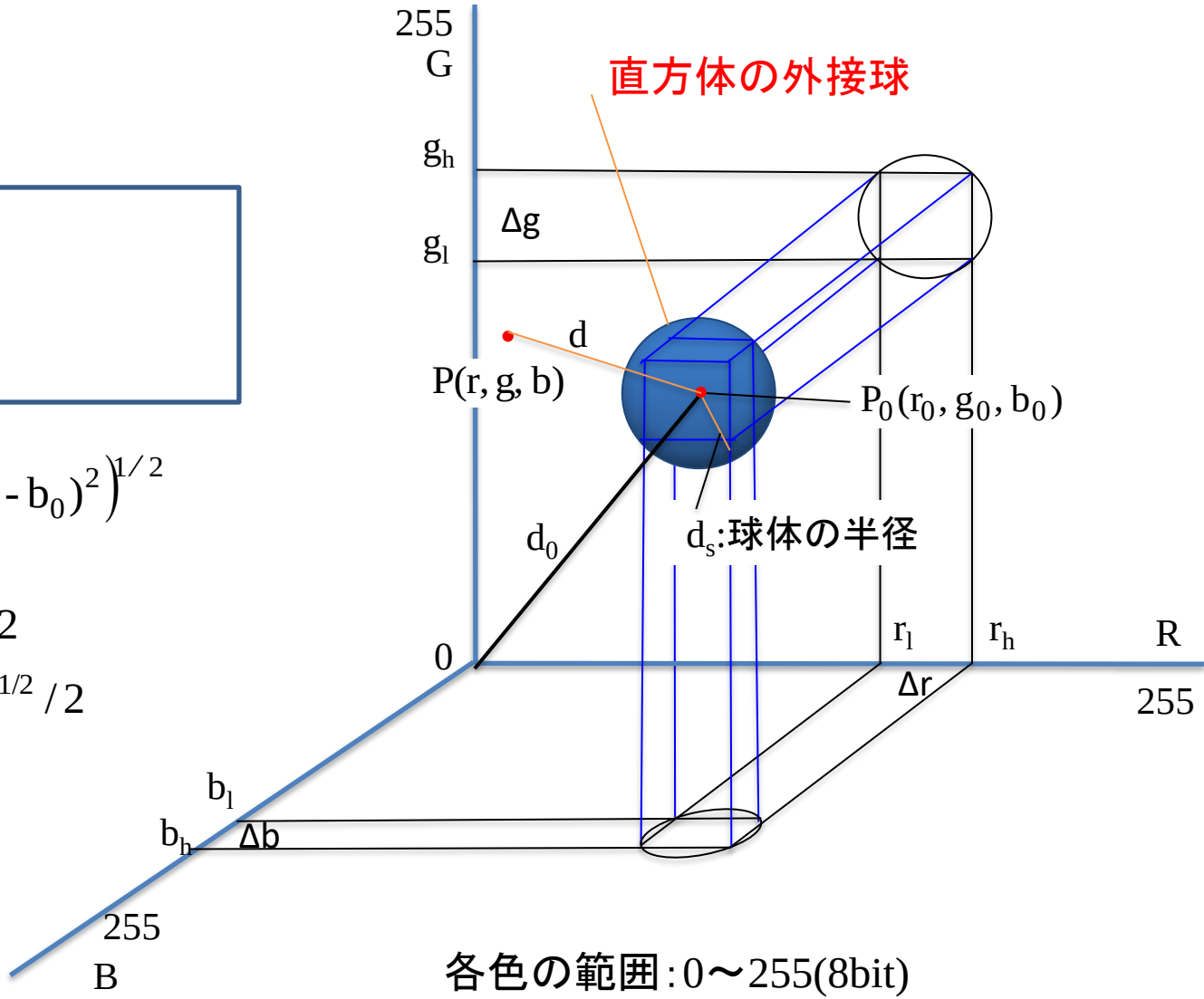
球体による色距離(Color distance) d

$$\Delta r = r_h - r_l$$
$$\Delta g = g_h - g_l$$
$$\Delta b = b_h - b_l$$

delr = redh – redl
delg = greenh – greenl
delb = blueh – bluel

$$d = \left((r - r_0)^2 + (g - g_0)^2 + (b - b_0)^2\right)^{1/2}$$

$$d_s = (\Delta r^2 + \Delta g^2 + \Delta b^2)^{1/2} / 2$$
$$= (delr^2 + delg^2 + delb^2)^{1/2} / 2$$



各色の範囲:0～255(8bit)
直方体の体積＜球体体積

Color distance

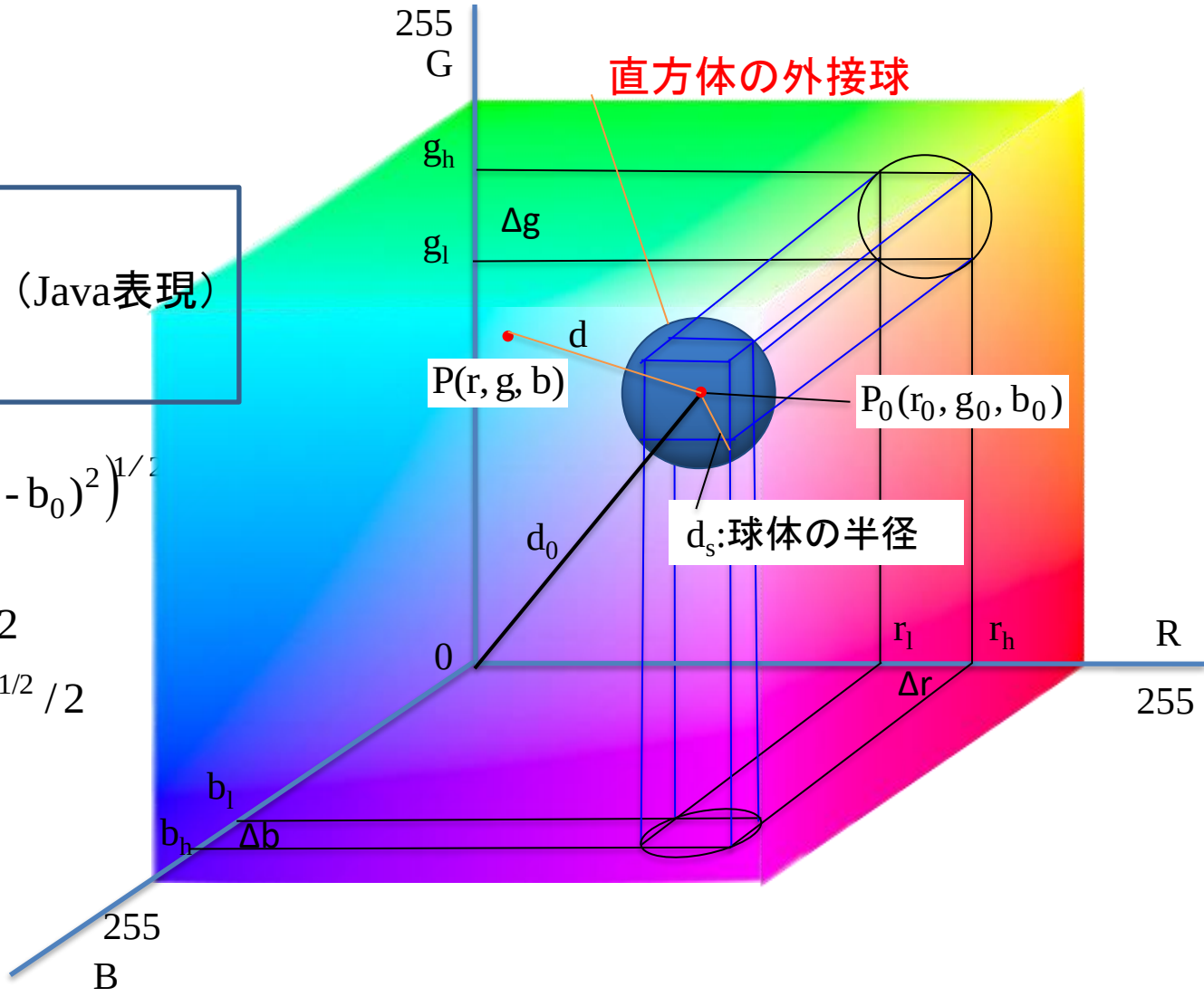
球体による色距離(Color distance) d

$$\Delta r = r_h - r_l$$
$$\Delta g = g_h - g_l$$
$$\Delta b = b_h - b_l$$

```
delr = redh – redl
delg = greenh – greenl        （Java表現）
delb = blueh – bluel
```

$$d = \left((r - r_0)^2 + (g - g_0)^2 + (b - b_0)^2\right)^{1/2}$$

$$d_s = (\Delta r^2 + \Delta g^2 + \Delta b^2)^{1/2} / 2$$
$$= (delr^2 + delg^2 + delb^2)^{1/2} / 2$$



Color distanceを使ったクロマキー処理

クロマキー（色抜き）の指定色範囲の中央位置を$P_0(r_0, g_0, b_0)$,

処理対象の画素位置を$P(r, g, b)$とすると,

P_0とPのColor distance dは

$$d = \left((r - r_0)^2 + (g - g_0)^2 + (b - b_0)^2\right)^{1/2}$$

$$r_0 = (rl + rh)/2,\ g_0 = (gl + gh)/2,\ b_0 = (bl + bh)/2$$

他方，指定色の範囲は

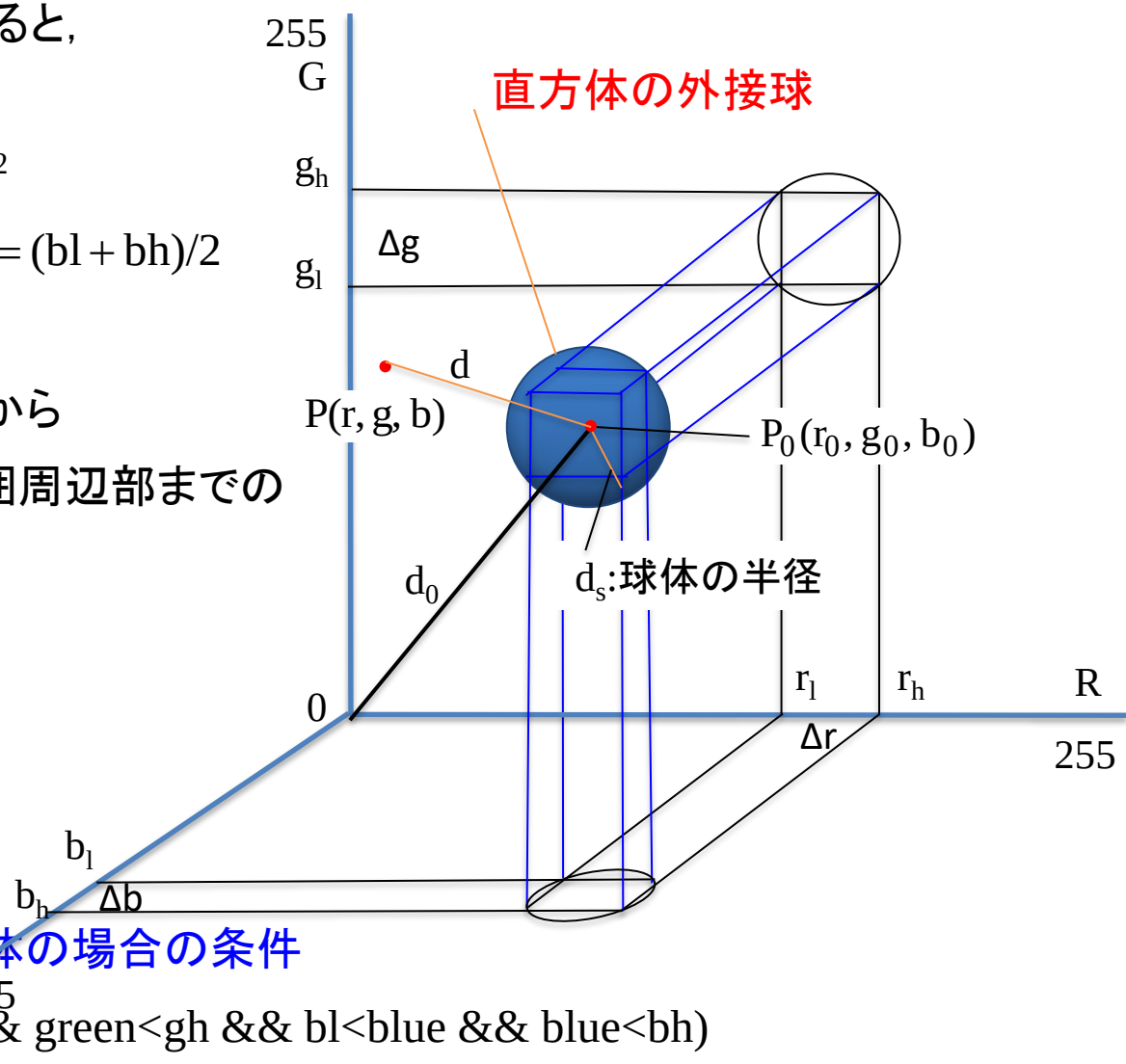
delr=rh-rl, delg=gh-gl, delb=bh-bl だから

中央位置$P_0(r_0, g_0, b_0)$から色指定範囲周辺部までの

色距離d_sは

$$ds = (delr^2 + delg^2 + delb^2)^{1/2} / 2$$

となる。



このため，クロマキーの条件は直方体の場合の条件

if (rl<red && red<rh && gl<green && green<gh && bl<blue && blue<bh)

を「d<dsの場合には表示しない」の条件に変えればよい。

Color distanceを使ったクロマキー処理

なお，平方根の計算にはMath.**を使う。
例えば $d = \left((red - r_0)^2 + (green - g_0)^2 + (blue - b_0)^2\right)^{1/2}$ は

d=Math.sqrt((red-r0)*(red-r0)+(green-g0)*(green-g0)+(blue-b0)*(blue-b0));

のように書く。

⇒ 例としてtest17.jpgを使ってクロマキー処理をしてみよう。

課題1：色空間での直方体と球体の条件比較

画像test17.jpgの例

直方体

redl<red && red<redh && greenl<green && green<greenh && bluel<blue && blue<blueh

球体

d<ds

d=Math.sqrt((red-r0)*(red-r0)+(green-g0)*(green-g0)+(blue-b0)*(blue-b0));

$$d_s = (delr^2 + delg^2 + delb^2)^{1/2} / 2$$

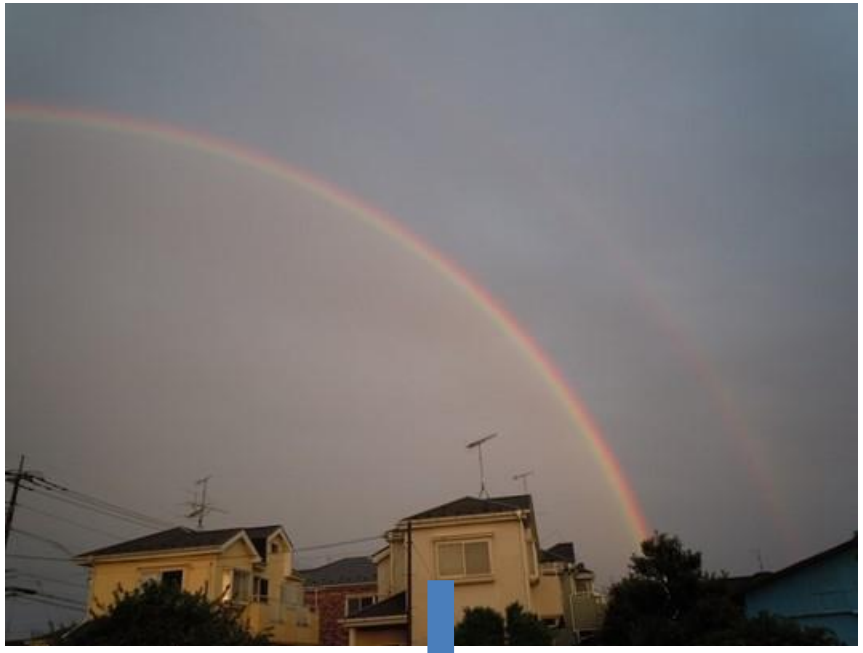
test5.jpgおよびtest18.jpgを使って色空間での直方体と球体の条件のクロマキー処理を比較し，違いの理由を理解する。

参考プログラム（球体によるクロマキー処理）

```
public void paint(Graphics g) {
        int ix,iy,i;
        int redl=24,redh=71,greenl=51,greenh=104,bluel=104,blueh=185;//blue flower
        int red,green,blue;//ピクセルの各色
        int r0,g0,b0,delr,delg,delb;
        int iw=1,ih=iw;//表示ピクセルのサイズを決める
        double d,ds;
        delr=redh-redl; delg=greenh-greenl; delb=blueh-bluel;
        r0=(int)(redl+redh)/2;g0=(int)(greenl+greenh)/2;b0=(int)(bluel+blueh)/2;
        ds=(Math.sqrt(delr*delr+delg*delg+delb*delb)/2);
        g.drawImage(img, 0, 0, this);    //元画像表示
        for (i=0;i<wh;i++){
                ix=(int)(i%w);
                iy=(int)(i/w);
                if (i>wh){
                        break;
                }
                red=(pix[i]>>16)&0xff;
                green=(pix[i]>>8)&0xff;
                blue=(pix[i]&0xff);
                d=Math.sqrt((red-r0)*(red-r0)+(green-g0)*(green-g0)+(blue-b0)*(blue-b0));
                if (d<ds){//カラー球体の内側か
                        //何もしない：クロマキー処理
                }else{
                        g.setColor(new Color(red,green,blue));
                        g.fillRect(ix+w+50,iy,iw,ih);
                }
        }
}
```

課題2：画像の埋め込み

test11.jpg



test17.jpg

クロマキー処理

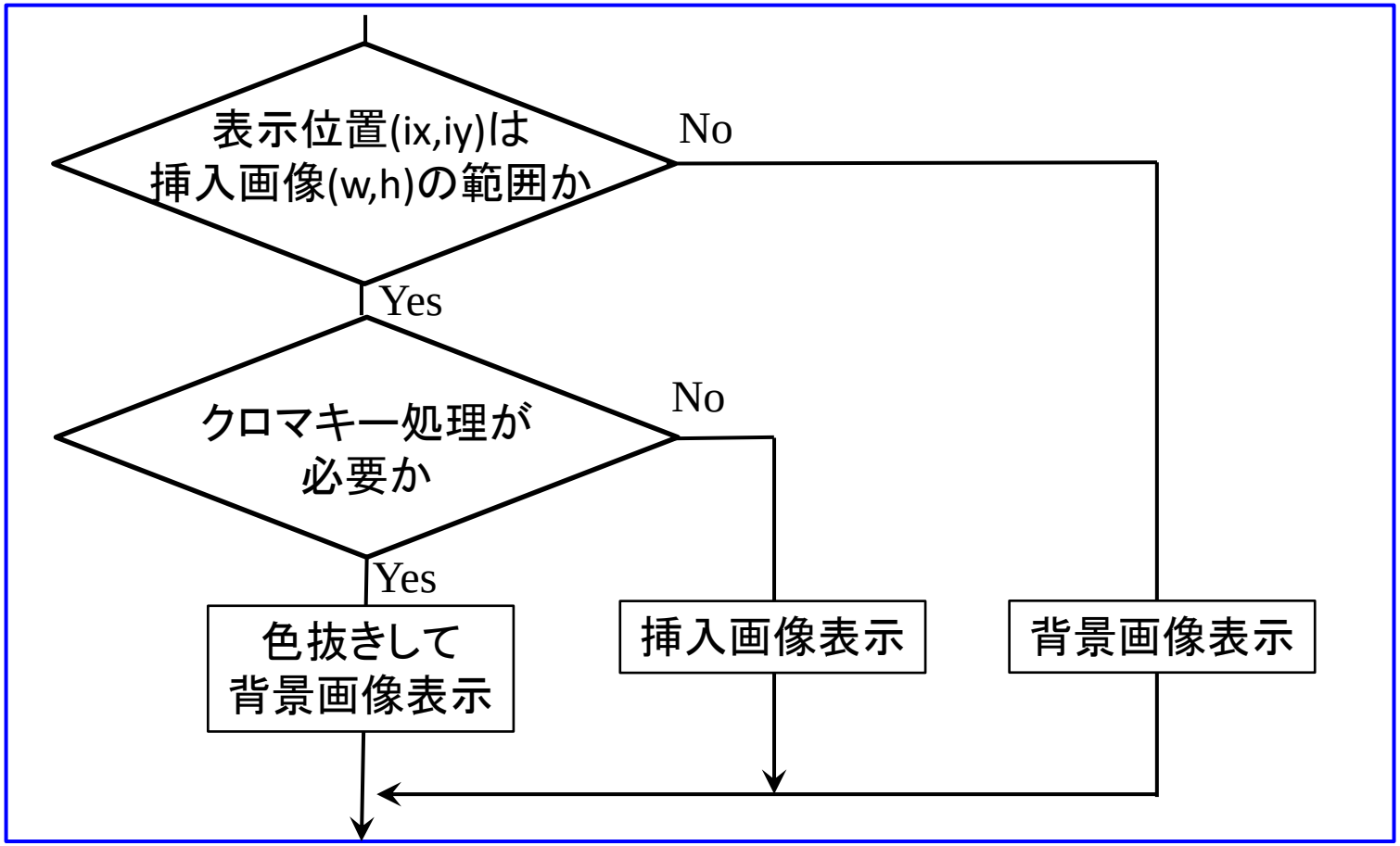
埋め込み

これまでの処理方法を応用する

- ○画像2枚（test11と15）の利用
- ○条件分岐
- ○変数増加

クロマキー処理で画像をはめ込む

フローチャート



参考プログラム

```
package gshori;

import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Toolkit;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.image.BufferedImage;
import java.awt.image.PixelGrabber;

import javax.swing.JFrame;

public class Jchromaf extends JFrame {// JFrame
	int pix[],pix1[],pixA[][],pixB[][];//インスタンス変数
	int w=0,h=0,wh,w1=0,h1=0,wh1;;
	int cnt=0;
	Image img, img1;  //読み込む画像のimageオブジェクト
	BufferedImage img2 =  null;
	PixelGrabber pixelG;//ピクセルを抽出するピクセルグラバーメソッドの宣言

	public static void main(String[] args) {
		Jchromaf frm = new Jchromaf();// フレームの生成
		frm.setSize(1800, 900);           // 窓サイズ横、縦
		frm.setVisible(true);             // フレームを表示する
	}

	public Jchromaf() {// コンストラクタ，戻り値がなくてもvoidがつかない
		setTitle("クロマキー画像挿入");// タイトル設定
		addWindowListener(new WindowAdapter() {
			public void windowClosing(WindowEvent e) {// 閉じるボタン対応
				System.exit(0);               // 終了する
			}
		 });
		 init();
		 //配列をイメージ化
		 img2 = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);//幅w高さhのBufferedImageの構築
		 img2.setRGB(0, 0, w, h, pix, 0, w);//開始点（0，0）から幅w高さhあるピクセル配列pixを幅wでスキャンしてセット
	}
```

参考プログラム

```
public void init(){//初期処理 init() メソッド
    int i,j,n,n1;
    int redl=37,redh=88,greenl=50,greenh=97,bluel=95,blueh=152;//blue doll
    int red,green,blue;//ピクセルの各色
    int r0,g0,b0,delr,delg,delb;//各色の半径と幅
    int x1=100,y1=320;//画像挿入開始点
    double d,ds;
    //背景画像
   Toolkit tk;//Toolkitメソッド
   tk = Toolkit.getDefaultToolkit();
   img = tk.getImage("images/test11.jpg");
   MediaTracker mt=new MediaTracker(this);//データの読み込み中に勝手に出力などをさせないメソッド
    mt.addImage(img,0);  //メディアトラッカーする上での対象のデータと配列0
    try{                 //例外を処理をする構文
          mt.waitForID(0); //例外が発生しそうなプログラム
    } catch (InterruptedException e){}    //例外の場合の処理
    w=img.getWidth(this);
    h=img.getHeight(this);
    System.out.println ("w="+w+"   h="+h);
    wh=w*h;
    pix=new int[w*h];    //読み込む画像の描画領域を一次配列を準備
    pixA=new int[w][h];//2次元配列の画像領域を設//
    pixelG=new PixelGrabber(img,0,0,w,h,pix,0,w);//pix配列に画像をセット
    try{          //例外を処理をする構文
          pixelG.grabPixels();   //例外が発生しそうなプログラム(ピクセルグラバー)
    } catch(InterruptedException e) {}  //例外の場合の処理
    for (j=0; j<h; j++){//背景画像の2次元配列変換
          for (i=0; i<w; i++){
                n=j*w+i;
                pixA[i][j]=pix[n];
          }
    }
    //挿入画像
   Toolkit tk1;//Toolkitメソッド
   tk1 = Toolkit.getDefaultToolkit();
   img1 = tk1.getImage("images/test15.JPG");
   MediaTracker mt1=new MediaTracker(this);//画像が読み込み完了するまで監視
    mt1.addImage(img1,0);
    try{                 //例外を処理をする構文
          mt1.waitForID(0); //例外が発生しそうなプログラム
    } catch (InterruptedException e){}    //例外の場合の処理
    w1=img1.getWidth(this);//読込み画像の横サイズ
    h1=img1.getHeight(this);//読込み画像の縦サイズ
    System.out.println ("w1="+w1+"   h1="+h1);
    wh1=w1*h1;//whは画素数の最大値
    pix1=new int[wh1];   //読み込み画像の配列pixの領域
    pixB=new int[w1][h1];
    pixelG=new PixelGrabber(img1,0,0,w1,h1,pix1,0,w1);//ピクセルグラバーメソッドの配列生成
```

参考プログラム

```
        try{          //例外処理をする構文
                pixelG.grabPixels();   //例外が発生しそうなプログラム
        } catch(InterruptedException e) {}  //例外処理の割り込み
        for (j=0; j<h1; j++){//挿入画像の2次元配列変換
                for (i=0; i<w1; i++){
                        n1=j*w1+i;
                        pixB[i][j]=pix1[n1];
                }
        }
        //クロマキー処理
        delr=redh-redl; delg=greenh-greenl; delb=blueh-bluel;
        r0=(int)(redl+redh)/2;g0=(int)(greenl+greenh)/2;b0=(int)(bluel+blueh)/2;
        ds=(Math.sqrt(delr*delr+delg*delg+delb*delb)/2);

        for (j=0;j<h;j++){//背景画像全体のループ
                for (i=0;i<w;i++){
                        if (j>h){
                                break;
                        }
                        //
                        if (i>x1&&i<(x1+w1)&&j>y1&&j<(y1+h1)){//クロマキー処理画像の範囲か
                                //i0=(iy)*w+ix;
                                red=(pixB[i-x1][j-y1]>>16)&0xff;
                                green=(pixB[i-x1][j-y1]>>8)&0xff;
                                blue=(pixB[i-x1][j-y1]&0xff);
                                d=Math.sqrt((red-r0)*(red-r0)+(green-g0)*(green-g0)+(blue-b0)*(blue-b0));
                                if (d<ds){////クロマキー処理が必要か
                                        //必要：背景画像のまま
                                }else{
                                        pixA[i][j]=pixB[i-x1][j-y1];//背景画像に挿入画像ピクセルを格納
                                }
                                }else{
                                        //背景画像のまま
                        }
                }
        }
        //2次元配列から一次元配列への処理
        for (j=0; j<h; j++){
                for (i=0; i<w; i++){
                        n=j*w+i;
                        pix[n]=pixA[i][j];
                }
        }
    }

    public void paint(Graphics g) {
         // 画像表示
        g.drawImage(img1, 10, 10, this);//挿入画像表示
        cnt++;
        System.out.println ("cnt="+cnt);
        g.drawImage(img2, w1+50, 10, null);//クロマキー処理結果の画像表示
    }
}
```