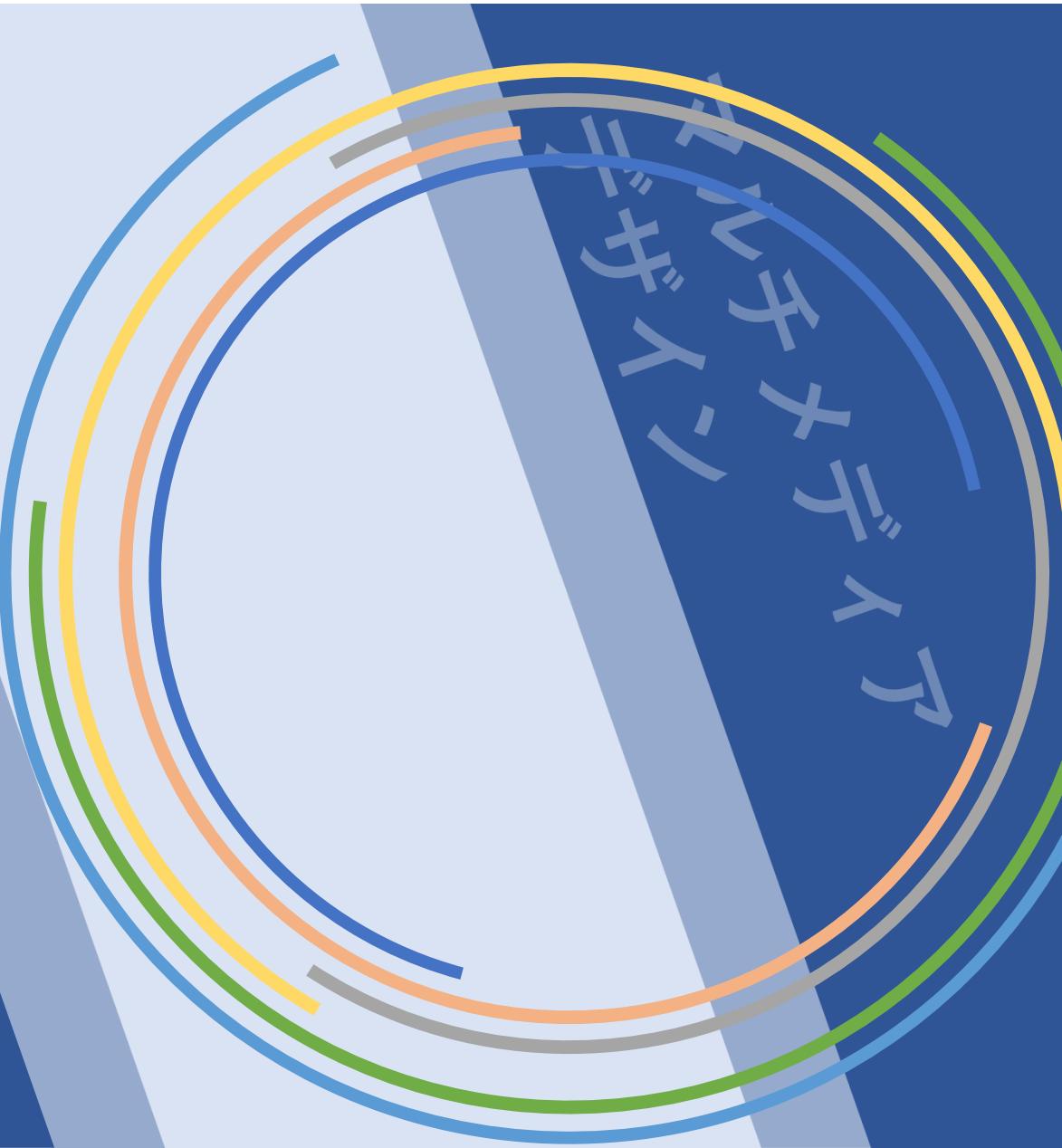


この授業のスライドはLMSで公開されます
実験の質問やレポートに関する質問などはSlackへ
<https://keio-st-multimedia.slack.com>



#6 データ表現

担当： 西 宏章





画像

- 画像情報処理とは

- 画像生成
- 画像表示
- 画像伝送
- 画像記録、蓄積
- 画像検索
- 画像変換
- 画像認識、記述
- CG

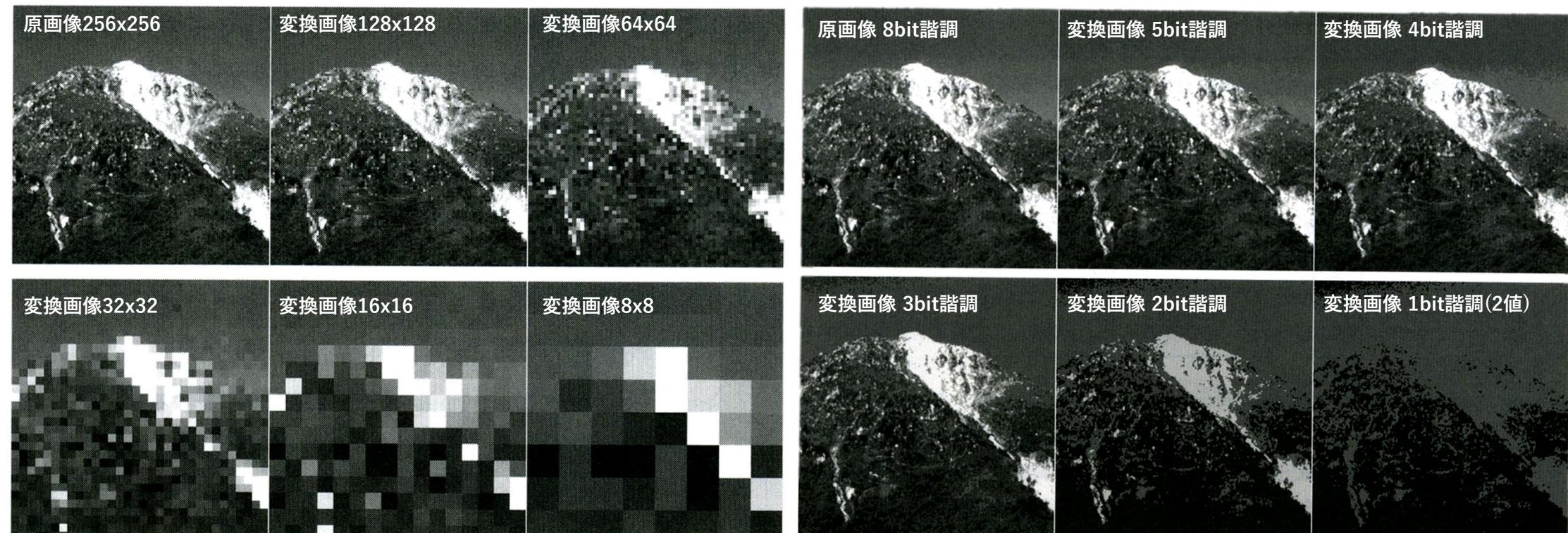




画像処理（標本化処理・量子化処理）

4

- 画像の保存
 - 通常は2次元配列に保存
 - int mono[100][100]; int color[100][100][3];
- 画素の粒度による差：標本化処理
- 画素の明度による差：量子化処理





量子化の違いによるカラー画質への影響

5

- ・パレットに従えば、減色しても比較的自然に見える
 - ・n色パレット：n個の色を使えるという意味で、各色の色空間は規定していない
 - ・自然な画像を少ない情報量で表現可能



原画像24bitフルカラー



2値化画像



2bit4色パレット



3bit8色パレット



4bit16色パレット



6bit64色パレット



8bit256色パレット

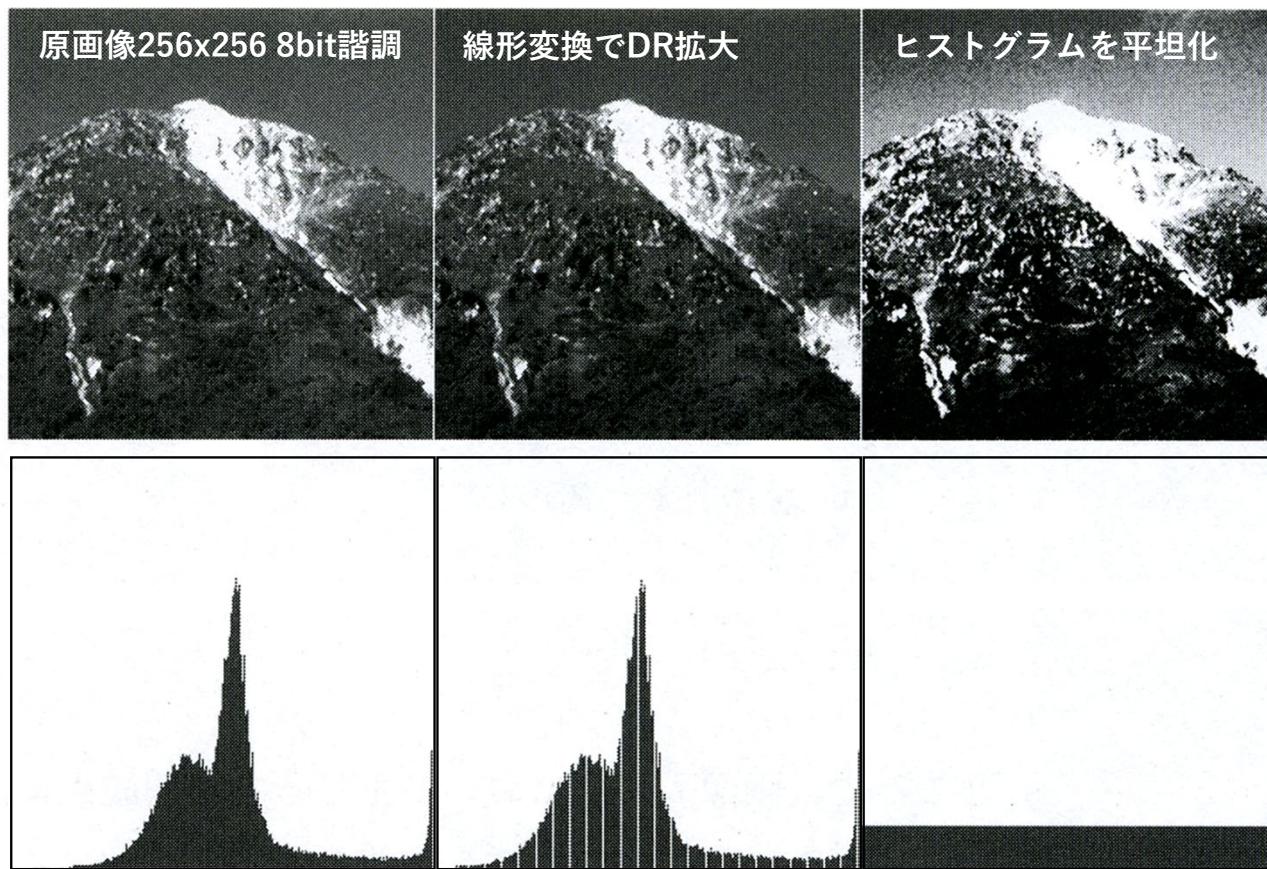




画像処理（濃度変換）

- コントラスト強調

- 画像が持つ濃度の変化幅(ダイナミックレンジ)を表現可能な階調幅全域に広げる
- 濃度値を線形や非線形に変換する処理や、濃度ヒストグラムの平坦化処理を施す

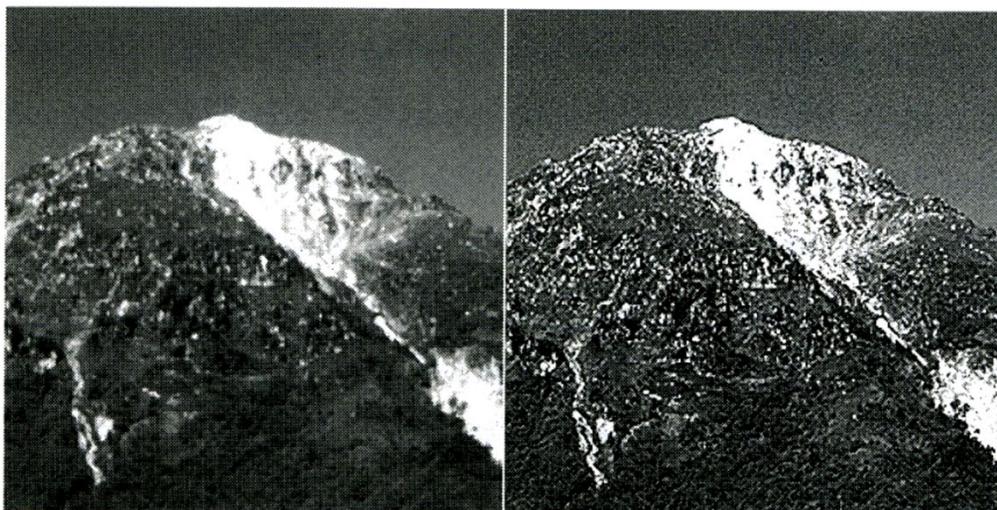




鮮鋭化処理

- 濃度変換の一種で、画像の印象を高める変換
- 明るさや色相が変化する部位では、その差が強調されて認識される
 - 心理学でいうマッハ効果
- この効果を利用し、鮮明な画像に加工する
 - 濃度勾配（ラプラシアン）を画素値に加える（引く）する

マッハ効果：色の境界部分では隣接する色の影響を受けて、一方は明るく、一方は暗く見える。明度段階の数を多くするか、境界部分に白または黒の線をいれると弱められる。





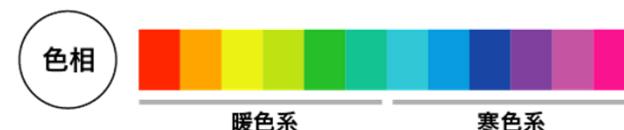
色相・明度・彩度

- 色度座標および3刺激値で表現され、外見を支配する要素、例えば光沢・透明度合い・地肌が異なれば、感覚的に色の差を感じ取る

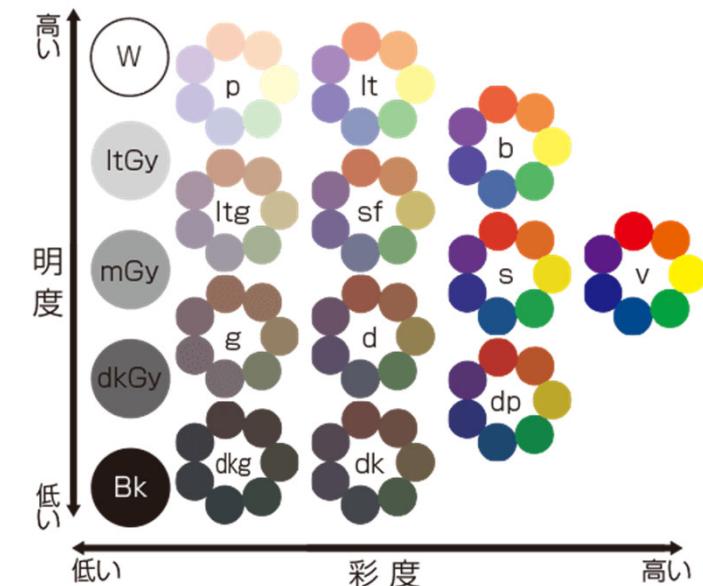
- 色相



- 彩度



- 明度



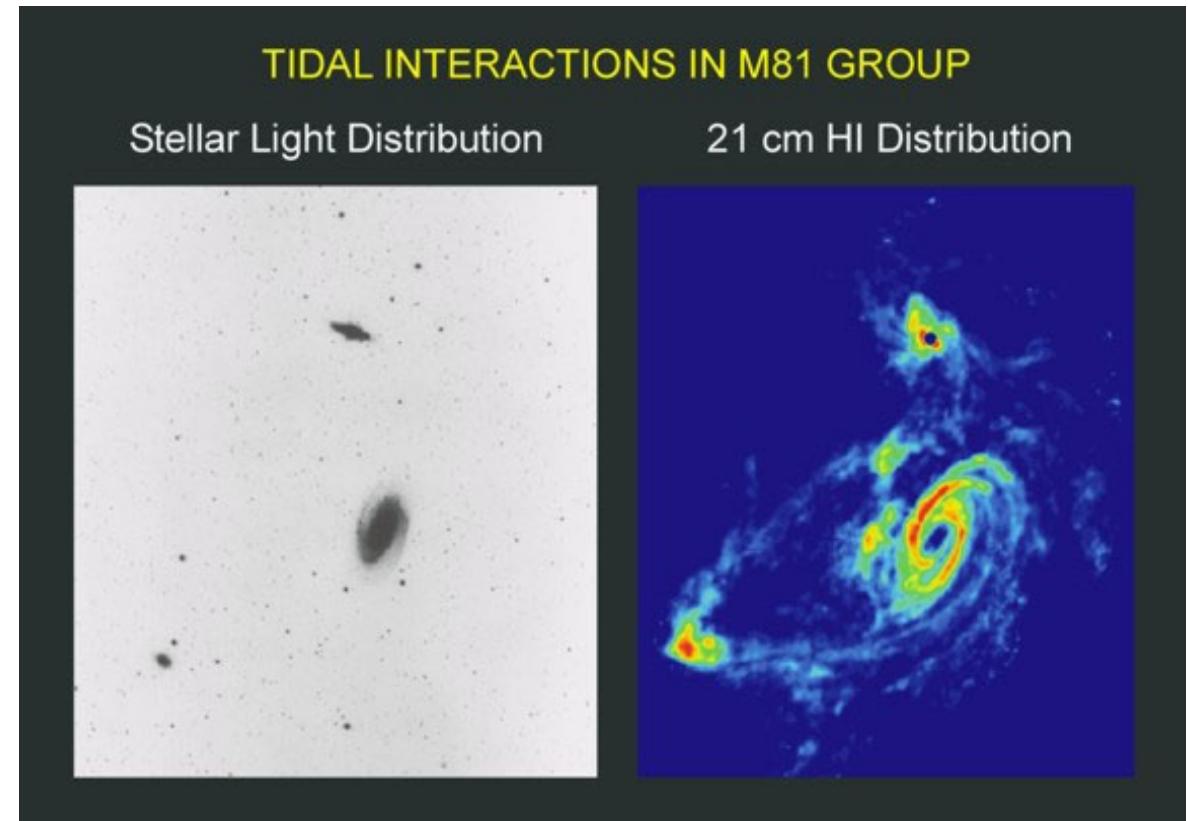
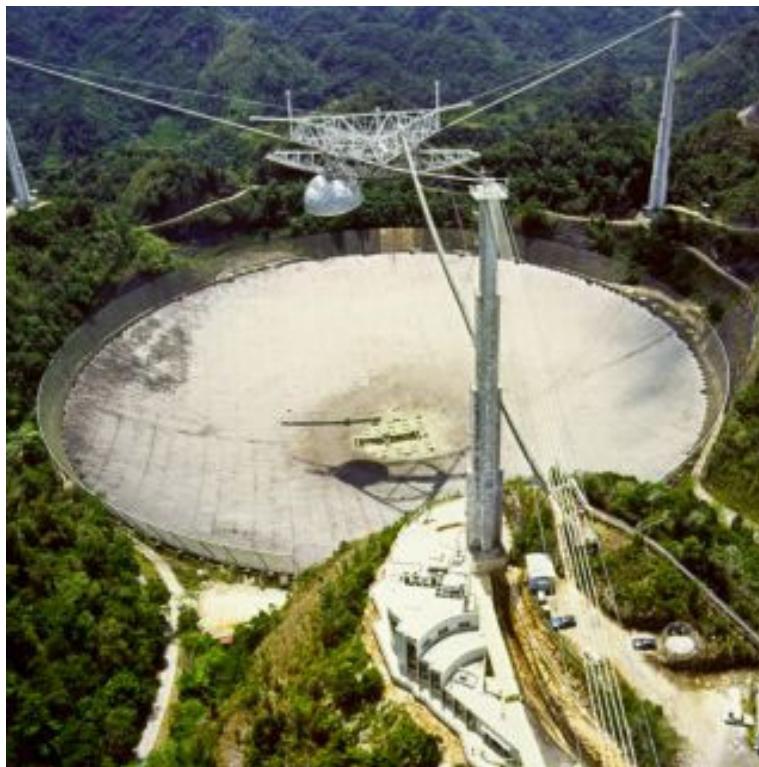
- 彩度変化よりも明度変化に敏感であり、より色相を際立たせる明度と彩度であれば、色相変化の感覚的な「差異」を最も大きく感じ取る





疑似カラー表示

- モノクロ画像の各画素を、照度値に応じて異なる色に変換
- 人間が濃度の変化より色の変化（色相）に対して敏感に感じ取ることができるため、画像解析に用いられる

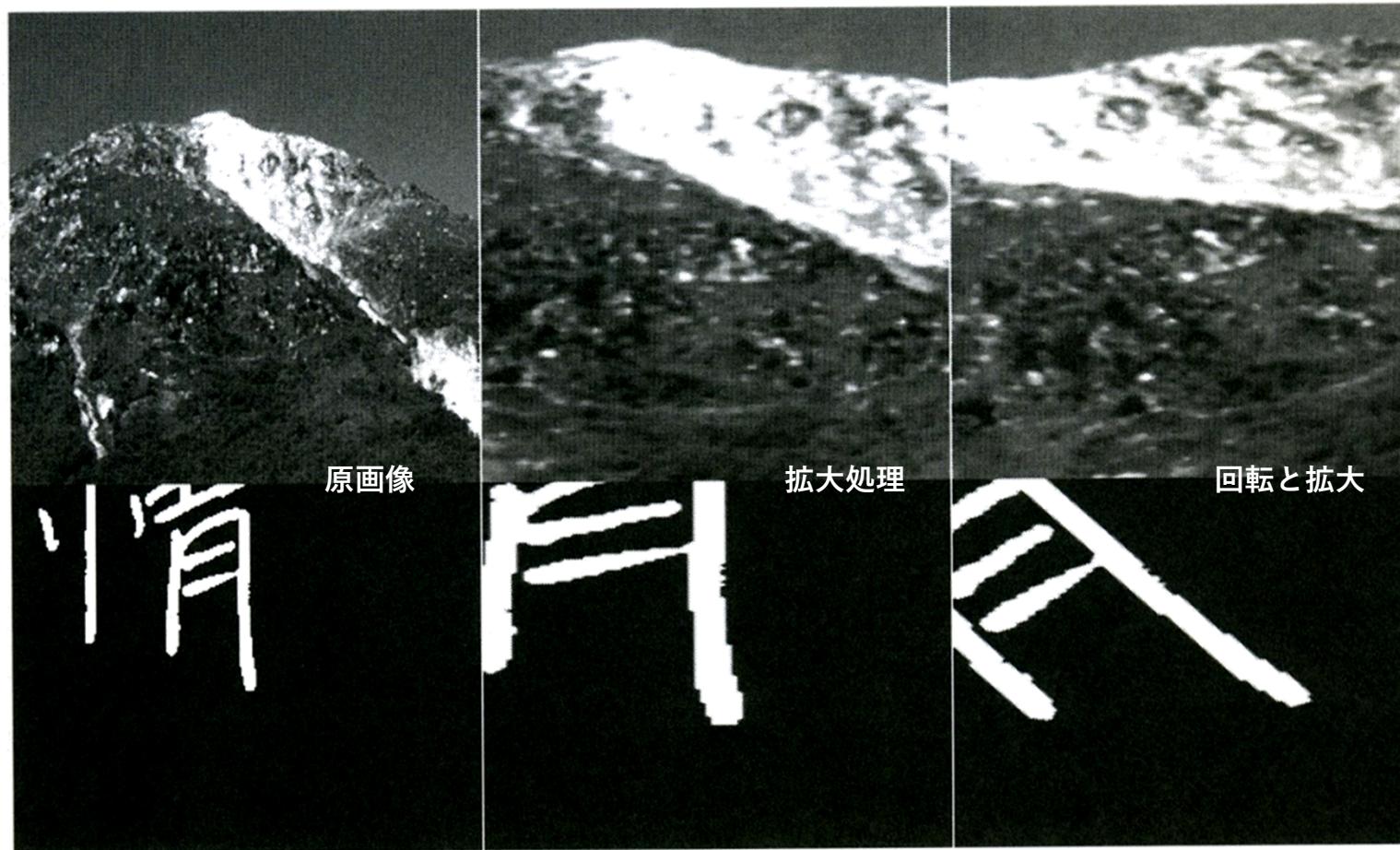




幾何学変換

10

- ・拡大、縮小、回転、並行移動などの変換を施す
- ・一次変換やアフィン変換を用いる





ProcessingとOpenCVによる実習

11

- Processingをインストールする
 - <https://processing.org/> へアクセス
 - Downloadから、Windows用、Linux用、MacOSX用を環境に合わせて導入
 - Windowsは特にインストールという作業はなくzipフォルダを展開してバイナリを実行するだけ
 - MacOSもほぼ同様
 - OpenCVを導入する
 - スケッチ(Sketch)から→ライブラリをインポート(Import Library)→ライブラリを追加(Add Library)→OpenCVを検索→OpenCV for Processingを導入
 - なお、MacOS Catalinaでは問題があり、processing Videoライブラリが利用できないという問題が報告されている（次のページの方法で解決する。授業には関係しない）
 - サンプルコードで様々な体験できる
 - processing.videoライブラリが必要な例なども存在するので注意
- サンプルプログラムを動かしてみる
 - ファイル(File)からサンプル(Examples)を選んでソースを呼び出し、▶ボタンを押すとコンパイルと実行が行われる
 - 実行してみよう

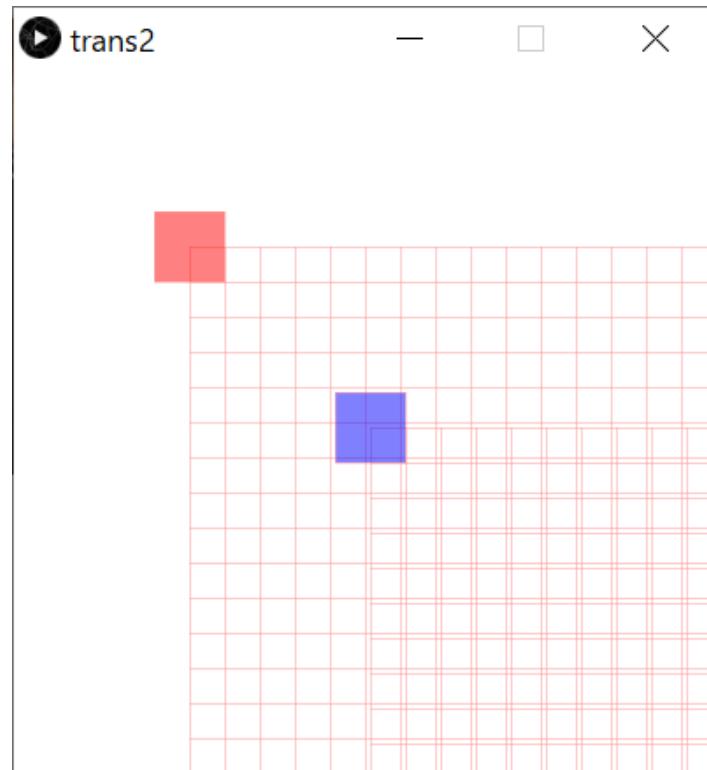




Processing: 座標の移動(平行移動変換)

12

- 座標を移動させてみる
 - drawgrid()はグリッドと四角■を描く関数
 - 座標と色を変えて2つ描画



```
void setup() {  
    size(400, 400);  
    rectMode(CENTER);  
    background(255);  
    translate(100, 100);  
    fill(255, 0, 0, 127);  
    drawgrid();  
    fill(0, 0, 255, 127);  
    translate(103, 103);  
    drawgrid();  
}  
void drawgrid() {  
    stroke(255, 150, 150, 127);  
    for (int x = 0; x <= 400; x += 20) {  
        line(x, 0, x, 400);  
    }  
    for (int y = 0; y <= 400; y += 20) {  
        line(0, y, 400, y);  
    }  
    rect(0, 0, 40, 40);  
}
```

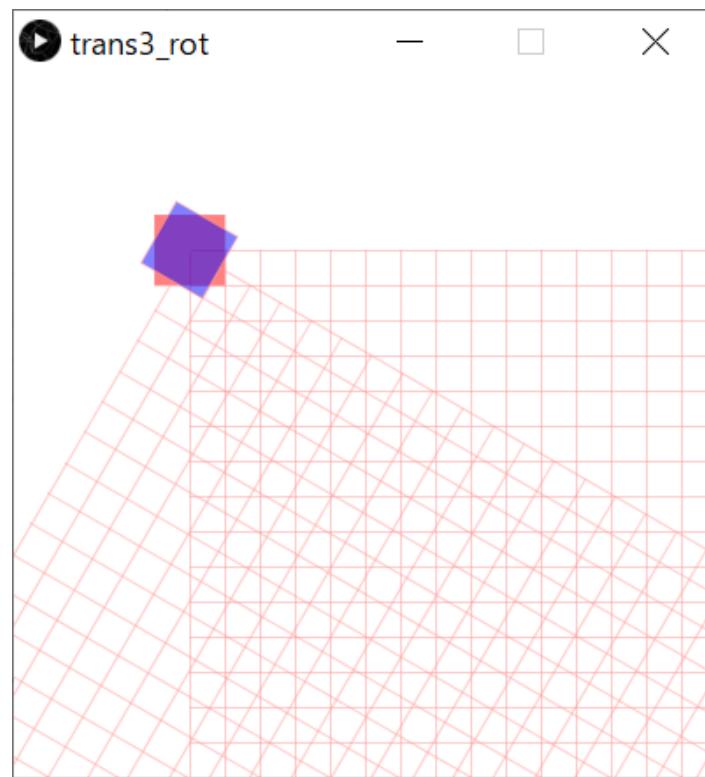




Processing: 座標の回転(回転変換)

13

- 座標を回転させるにはrotateを呼び出す
 - ここでは30度右回りに回転させている



```
void setup() {  
    size(400, 400);  
    rectMode(CENTER);  
    background(255);  
    translate(100, 100);  
    fill(255, 0, 0, 127);  
    drawgrid();  
    fill(0, 0, 255, 127);  
    rotate(radians(30));  
    drawgrid();  
}  
void drawgrid() {  
    stroke(255, 150, 150, 127);  
    for (int x = 0; x <= 400; x += 20) {  
        line(x, 0, x, 400);  
    }  
    for (int y = 0; y <= 400; y += 20) {  
        line(0, y, 400, y);  
    }  
    rect(0, 0, 40, 40);  
}
```

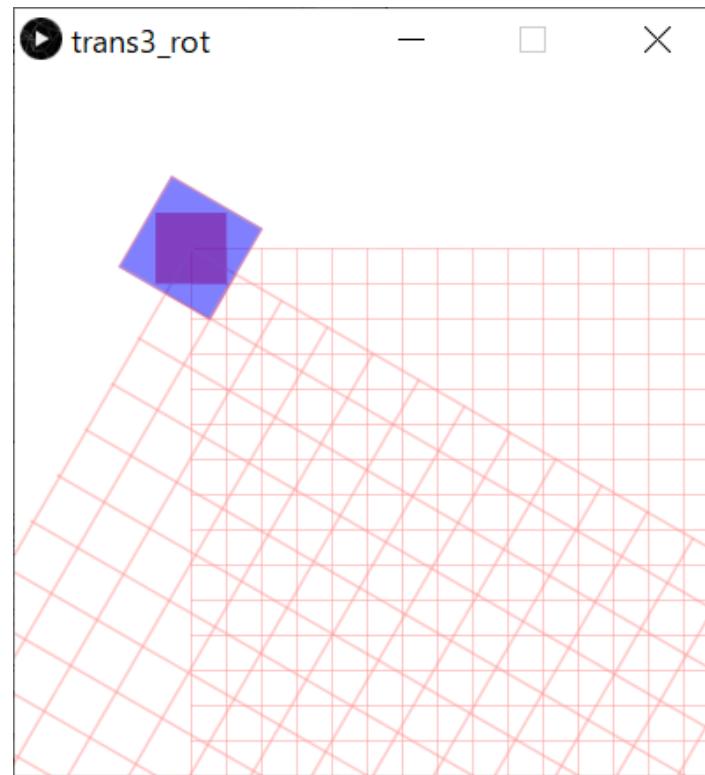




Processing: 座標の拡大・縮小(スケール変換)

14

- 座標の拡大・縮小にはscale関数を呼び出す
 - ここでは、30度右回りに回転したあと、1.5倍に拡大している
 - 0.5などと指定すれば縮小変換になる



```
void setup() {  
    size(400, 400);  
    rectMode(CENTER);  
    background(255);  
    translate(100, 100);  
    fill(255, 0, 0, 127);  
    drawgrid();  
    fill(0, 0, 255, 127);  
    rotate(radians(30));  
    scale(1.5);  
    drawgrid();  
}  
void drawgrid() {  
    stroke(255, 150, 150, 127);  
    for (int x = 0; x <= 400; x += 20) {  
        line(x, 0, x, 400);  
    }  
    for (int y = 0; y <= 400; y += 20) {  
        line(0, y, 400, y);  
    }  
    rect(0, 0, 40, 40);  
}
```





Processingの基本

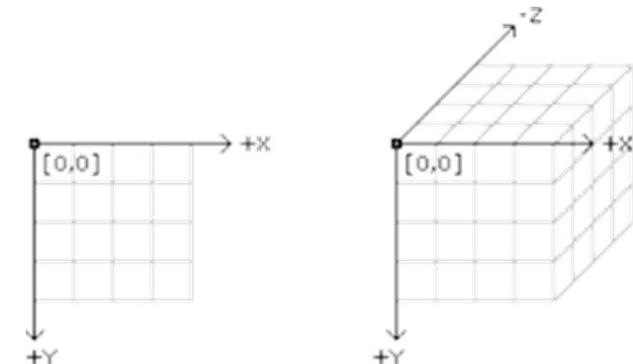
15

- `setup()`と`draw()`

- この2つの関数が基本
- `setup()`は最初に1度呼び出され、初期化に用いられる
- `draw()`は描画にある一定の周期やイベント発生時に呼び出され、描画に用いられる
 - `frameRate(r)`関数で秒間r回描画することを宣言する
 - `frameCount`変数により何フレーム目かを知ることができる
 - `background(R, G, B)`で背景を指定できる

- 様々な図形の描画関数がある

- 点: `point(x座標, y座標);`
- 四角: `rect(左上のx座標, 左上のy座標, 幅, 高さ);`
- 円・橢円: `ellipse(中心のx座標, 中心のy座標, 横直径, 縦直径);`
- 塗りつぶし
 - `fill(R, G, B);` 以降の図形描画を指定色で塗りつぶす
 - `nofill();` 塗りつぶし無し
- 線の描画も様々な手法があるので調べてみよう





演習問題（4）

16

問題(1)

- Processingを導入しなさい
 - いくつかバージョンがあるが、最新バージョンを導入するとよい
 - Processingは普通に使うなら何ら問題がないが…
 - ちょっと機能を強化しようとすると、いろいろと問題がある
 - 日本語を極度に嫌うアプリケーションであり、日本語は使うな
 - 日本語のディレクトリに入れるな(デスクトップもだめ)
 - バージョンが上がって改善しているが、ライブラリ導入時に問題が発生する可能性があり
 - カメラなどのデバイスを使うときや、ライブラリのインストール時は管理者権限で実行する
 - それでもダメなバージョンも存在する
 - 特にVideo系は、現状全滅状態にある
 - MacOS Catalinaでも不具合が報告されているが、改善報告もある
- 導入したら動作を確認しなさい
 - DEMO/Graphics/Graphicsにあるデモがお勧め
 - 特にplanetsや、Yellowtail(マウスで図形を描くと良い)がお勧め





- 次のようなプログラムを作成して動作させなさい

- setupでsize(500,500)、背景を白(255)とする
- drawで赤色で0,0から100,100の矩形の中に丁度納まる1辺80の正方形と直径80の丸を赤色で描画(2つとも収まっているればなんでもよい)
- 軸を(200,50)移動後、45度右回転し、1.5倍のスケールで同じ図形を緑色で描画
- 同じ変換をもう一度鉛越して青色で描画

注意事項

- これらの設問に対する回答を、Microsoft Wordファイルで作成
- LMSで提出すること
- A4で作成すること
 - なお、常識的な範囲でページ数を超過することは問題ありません
- ソースコード、動作画面をキャプチャして貼り付けること**
- 最初にタイトルとして「演習問題（4）」と書き、名前と学籍番号を記載すること
このフォーマットに従っていないレポート答案は受け取らない
- 締め切りなど詳細はLMSを確認すること



レポートに関する注意点

- レポートについては、必要事項だけ揃えて頂ければOKです
- 以下の内容は記載する必要はありません。記載しても点数に影響しません
 - 感想や数行に渡る過剰な説明文章
 - 課題の理論・課題の内容の書き直し・課題の感想
- ただし、「追加事項や、課題に対する工夫を行った」ということは、簡単に書いてください
 - 何か拡張したっぽいのだけど、何をしたのか理解するのが大変ですと点数に結びつきにくいです
 - 拡張は、相応に評価します
- かといって、拡張を無理にお願いするわけではありません。
- あくまでも授業内容の確認として課題レポートを出してあります
 - 何か凄いことをしないとまずい、などと考えるのは本末転倒です
 - 理解して、とにかく動いていればOKです
 - このコードまずいなあ、でも動いてるなあ、という場合でも、こちらとしてはOKとして採点します
 - コードの美しさや、実行効率などは、問いません
 - でも、最後にコードの整形ボタンを使って、整形ぐらいはしてもらえていると助かります
- 次のページで、Processingに関するコツについて説明します





Webで設計する場合

19

- <https://www.openprocessing.org> の利用がよさそうです
 - なお、授業内容のすべての動作を保証するものではありません。どうしても癖があります
 - おすすめ、サポートはあくまでも、Windows, MacOSのProcessingです
- まず、先に示した前提で設計してください
 - 授業中の参考例は、行数を減らすため、前提に従っていない場合も多々あります
- 画像の扱いは次の通りです
 - Webで動作するOpenProcessingは、アカウントを登録すればファイルが配置できます。
 - 画面右にSketch, Files, Editorとありますので、Filesを選択してください。
 - ここでファイルをドラッグせよと出ていれば、画像ファイルをドラッグして配置してください。
 - Saveしなさいといわれる場合は、一度Saveを押して保存します。保存にはアカウントの作成が必要です。
 - アカウント作成後、縦に点が3つならんでいるボタンを押すと、再びSketch, Files, Editorと出てきます。
 - ここでFilesをクリックすると、今度はファイルをドラッグせよ、とでてきます。画像ファイルをドラッグして配置してください。
 - この状態であればloadImage("ファイル名")で画像を読み込むことができます
 - さらにこの方法で読み込んだ画像はfilterなどの処理が可能です
 - オンラインのURLで指定した画像は、filterなどの処理ができません。
 - iPhone, iPadなどで動作するProcessing iCompilerは、ファイルを直接配置してもfilter処理ができません
 - これらの手段を使った画像処理を扱うProcessing記述は現状ではサポート対象外とさせてください
 - Androidについては未確認です
- Webをご利用の場合、特にsetupとdrawの記述におけるルールに忠実に従うようにソースコードを変更してください





Processingを利用する皆さんへの注意事項

20

- フォルダ名（ディレクトリ名）に注意してください
 - フルパスに日本語名称を入れると動作しない場合があります
 - 例えば、Windowsでは「デスクトップ」といった名称、やりがちなのは、「マルチメディアデザイン」といったフォルダ名、さらには、Userの下の「西」といった個人名に至るまで、様々トラップがあります。
 - Windowsユーザのおすすめは、C:\MDなど、Cドライブの直下にファイルを配置して実習することです。
 - もし動作しない場合は、フルパスのディレクトリ名を疑ってみてください。

