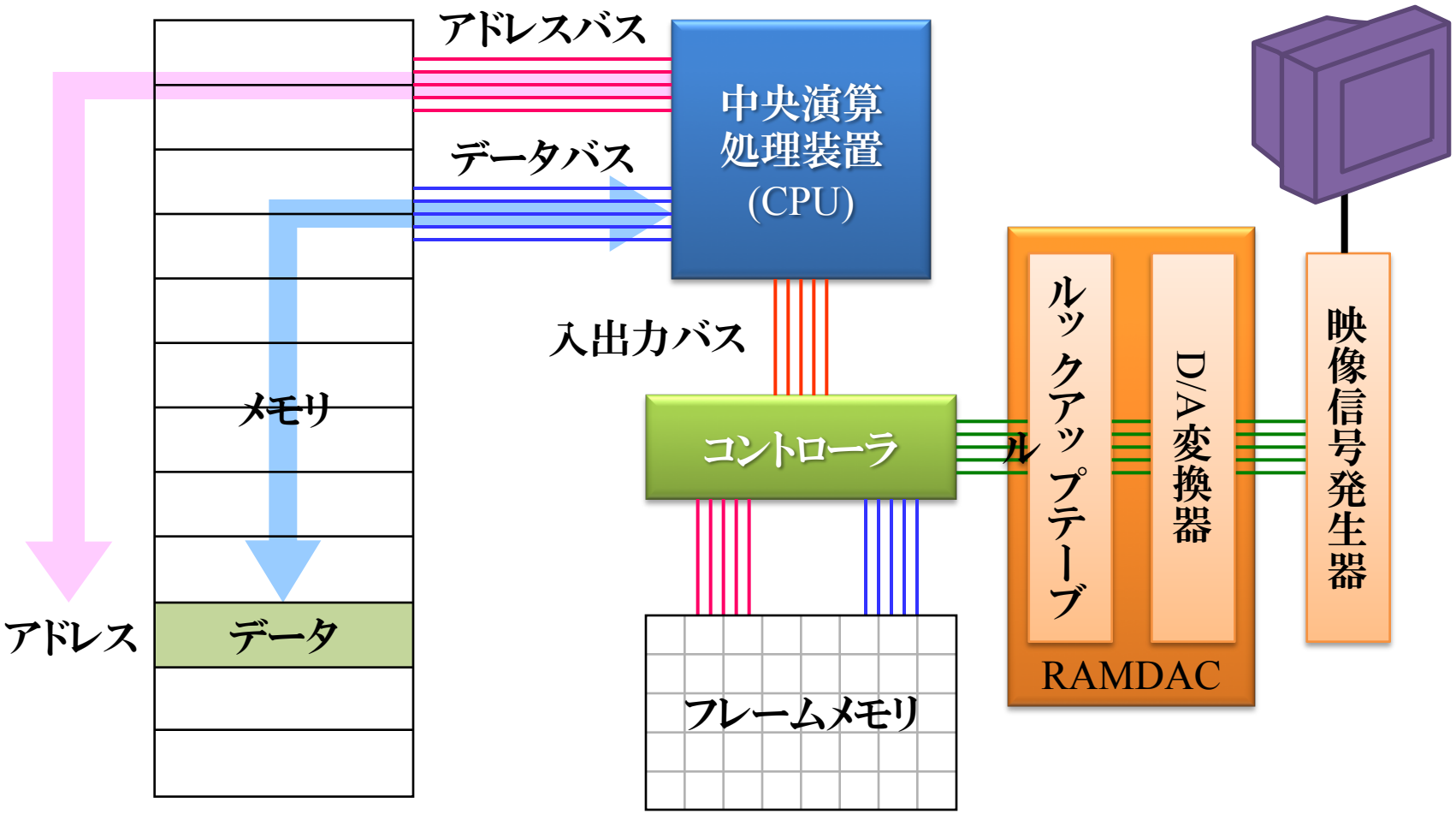
コンピュータグラフィックス

第2回：「画材」としてのコンピュータ

「画材」としてのコンピュータ

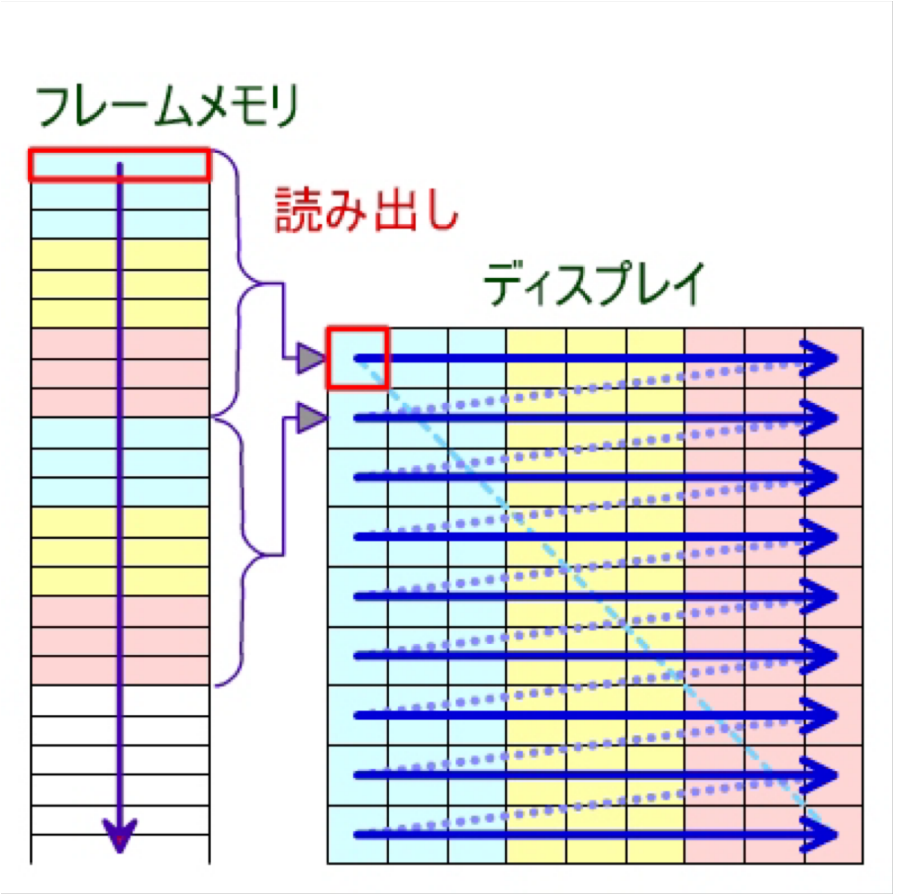
- データは絵の具
  - 明るさの値をデータとして用いる
  - データを光の強さに変換して色を再現する
- メモリはカンバス
  - メモリはデータを格納する空間
  - 色のデータをメモリ上に配置して形を表現する
- 色や形を自在に作ることができる
  - 計算で作ることもできる

コンピュータの構造


アドレスバス
データバス
中央演算
処理装置
(CPU)
入出力バス
メモリ
アドレス
データ
コントローラ
ルックアップテーブル
D/A変換器
RAMDAC
映像信号発生器
フレームメモリ

フレームメモリ

- 画像を記憶するメモリ
  - フレームバッファ
  - VRAM (Video RAM)
- 画像データの映像化
  - メモリを繰り返し読み出して，その内容をもとに映像信号を作る
- 走査線 (scan line)
  - 1行分のデータや画像

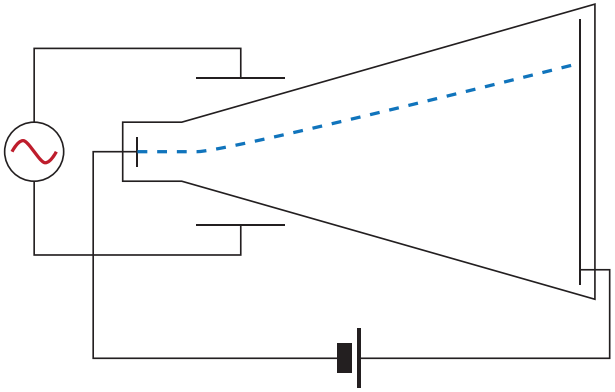


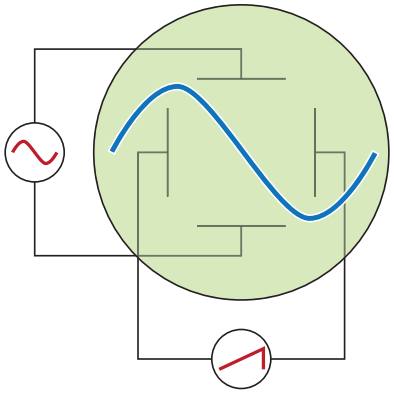
ラスターグラフィックス

- フレームメモリを前提にしたCG
  - 表示する画像は点（輝点）の集合
- 「ベクターグラフィックス」というのもある
  - 「点」ではなく「線」を表示できる装置を使う
    - 静電偏向CRT, レーザリアム, …
    - XYプロッタ, …
  - 初期のCGではこれが主流だった
    - 陰影（濃淡）を表現することが難しかった
    - 表示装置が「テレビ」となじまなかった

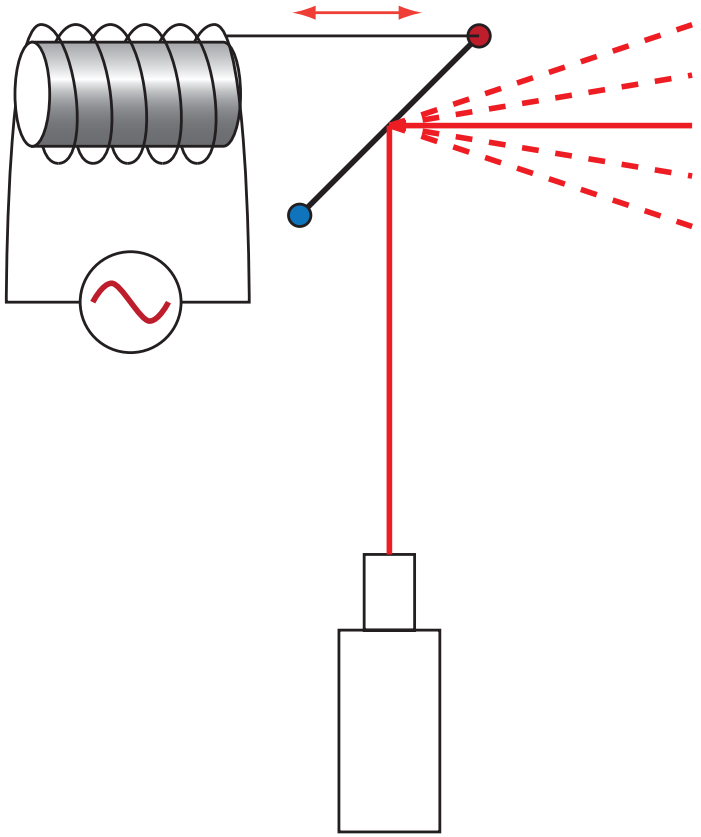
ベクターグラフィックスの例

静電偏向方式





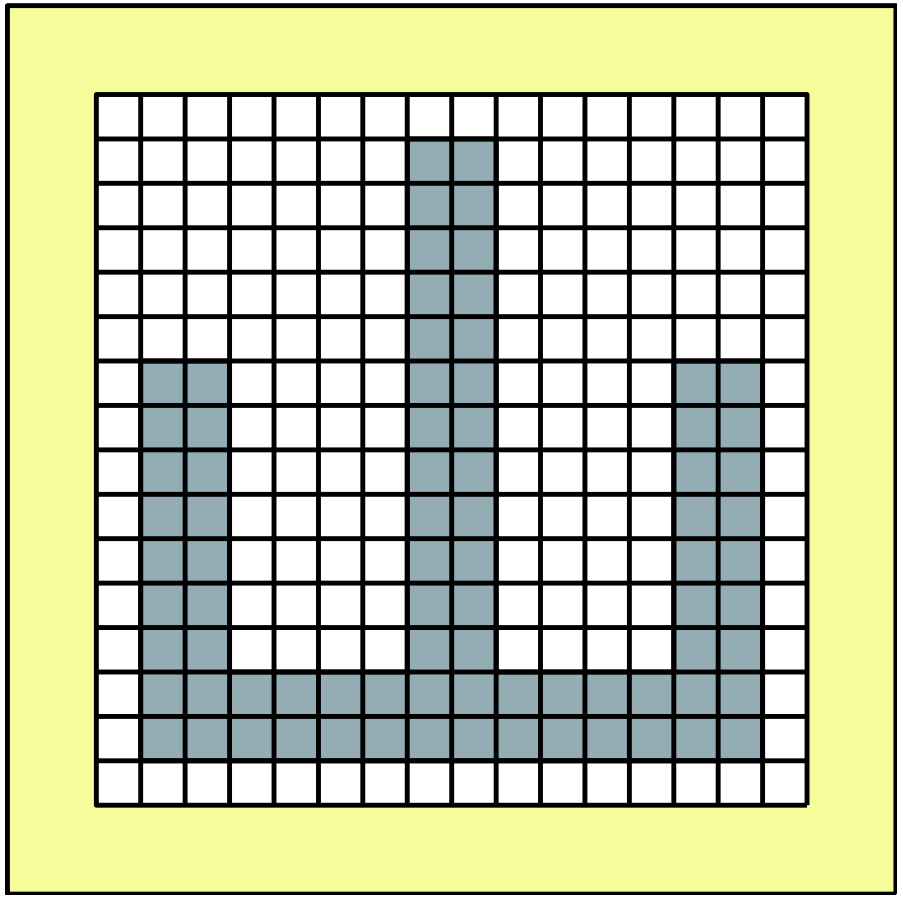
ボイスコイル方式



Ivan Sutherland と Sketchpad

デジタル画像

- 形を点の集合で表す
  - 画像を格子状に分割する
  - 一つ一つの点が濃度（色，明るさ）を表す
- この点を画素（pixel）と呼ぶ
  - 画像を構成する最小単位
- ラスター画像とも呼ばれる



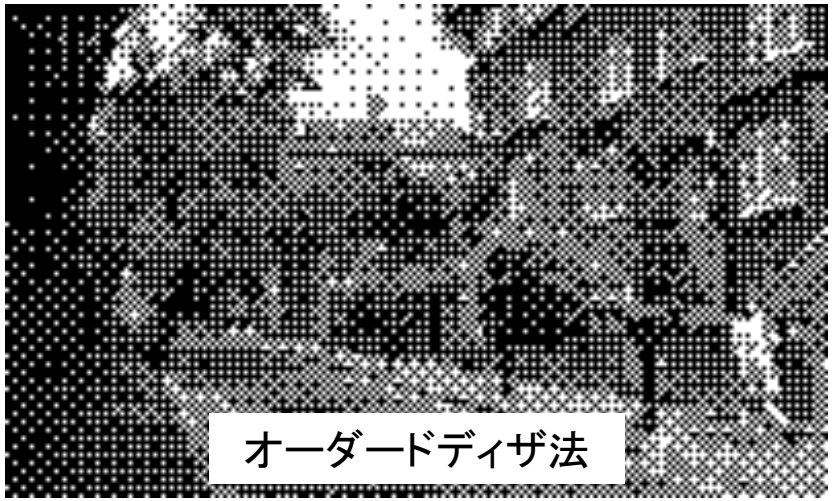
デジタル画像の種類

- 2値画像
- グレースケール画像
- フルカラー画像
- インデックスカラー画像

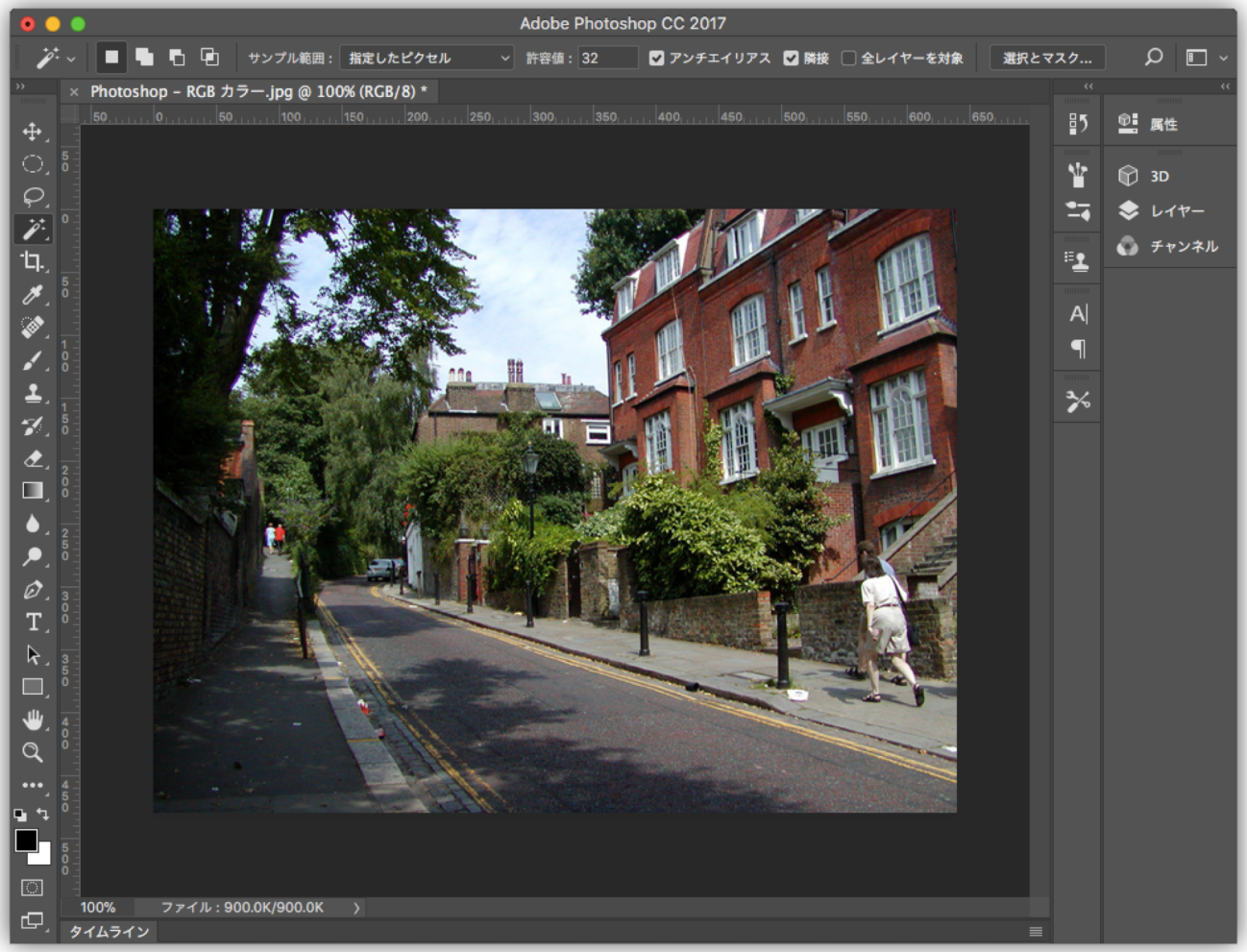
2値画像

- 1画素を1bitで表す
  - 白・黒などの2値
    - 白→0，黒→1 など
    - コピー，ファクシミリ等
- 濃淡は表現できない
  - 面積階調法を用いて表現
    - パターンディザ法
    - オーダードディザ法
    - ランダムディザ法
    - 濃度拡散法
    - 網点(新聞など)

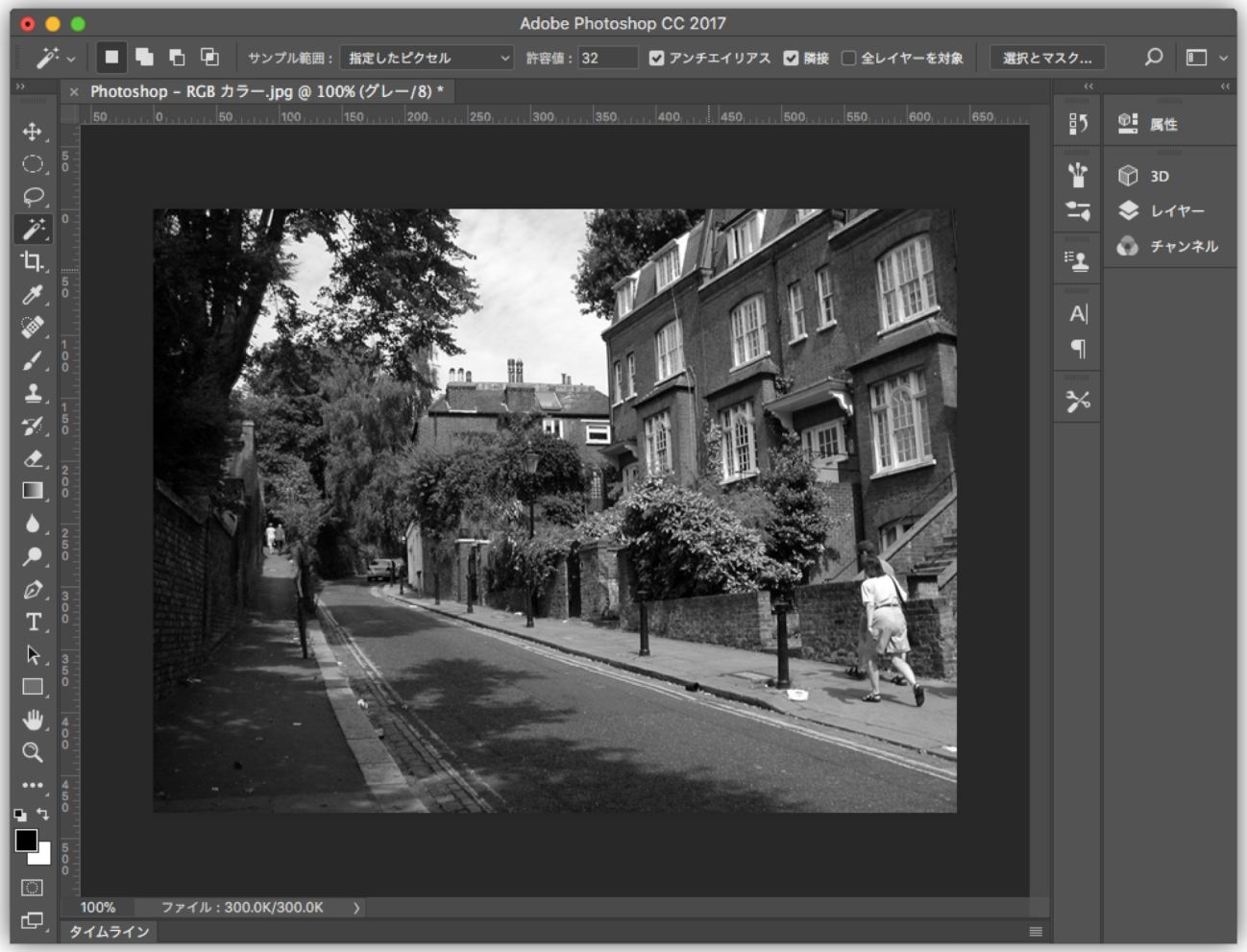
面積階調法



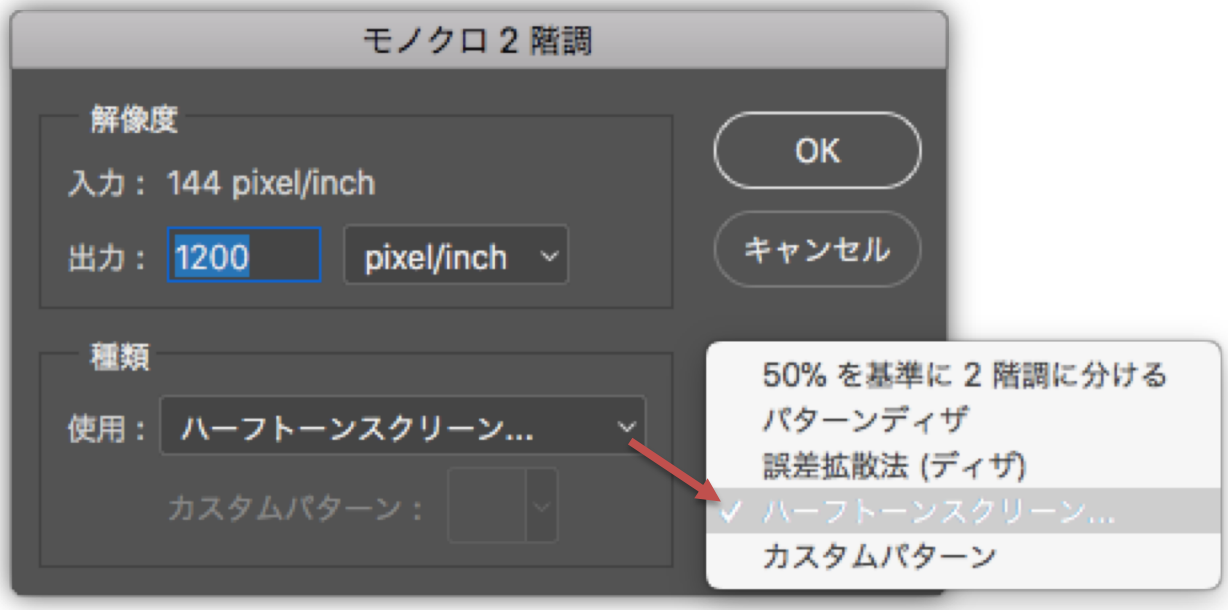
Photoshop


Adobe Photoshop CC 2017
サンプル範囲：
指定したピクセル
許容値：
32
アンチエイリアス
隣接
全レイヤーを対象
選択とマスク...
Photoshop – RGB カラー.jpg @ 100% (RGB/8) *
属性
3D
レイヤー
チャンネル
100%
ファイル：900.0K/900.0K
タイムライン

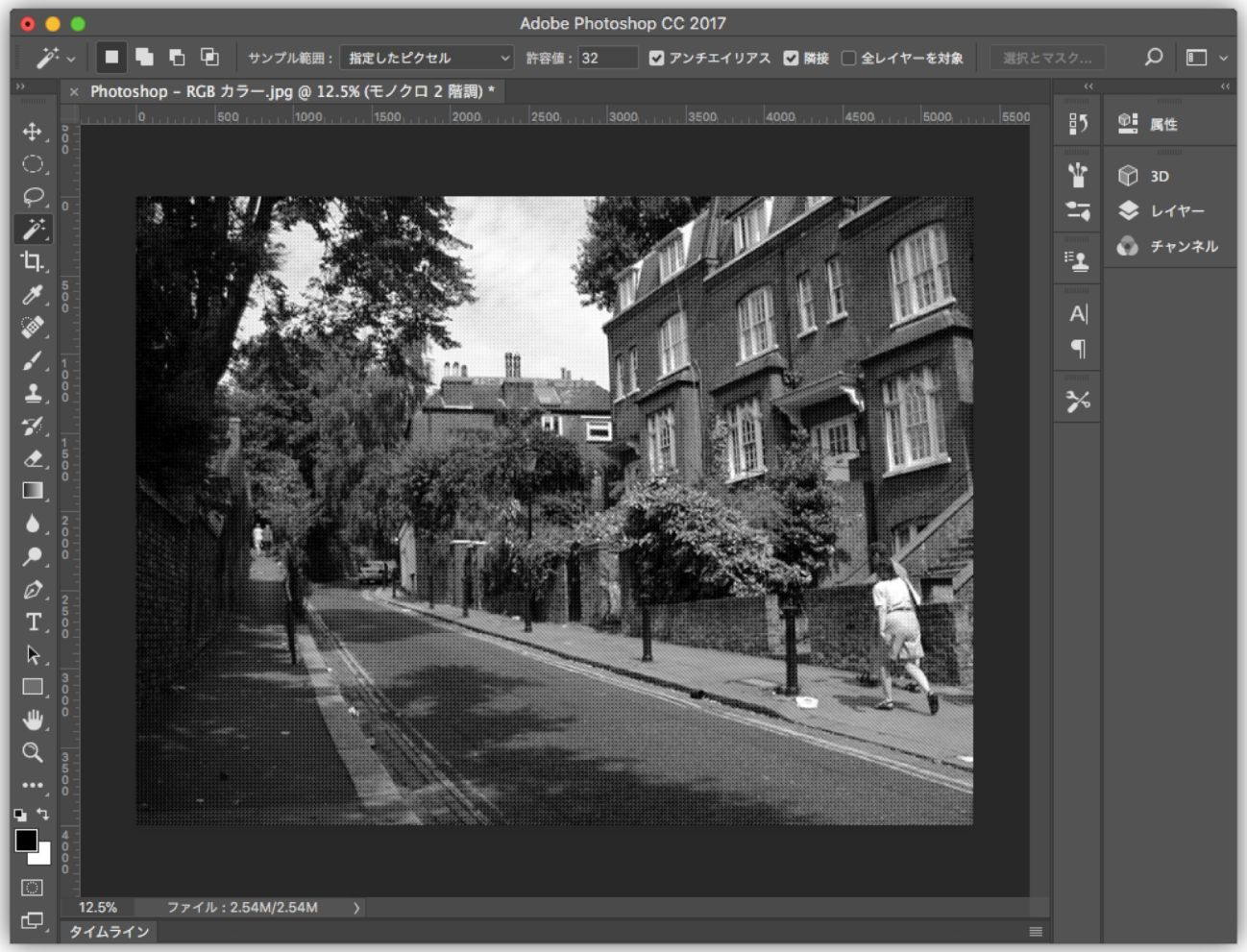
グレースケールに変換


Adobe Photoshop CC 2017
サンプル範囲：
指定したピクセル
許容値：
32
アンチエイリアス
隣接
全レイヤーを対象
選択とマスク...
Photoshop – RGB カラー.jpg @ 100% (グレー/8) *
属性
3D
レイヤー
チャンネル
100%
ファイル：300.0K/300.0K
タイムライン

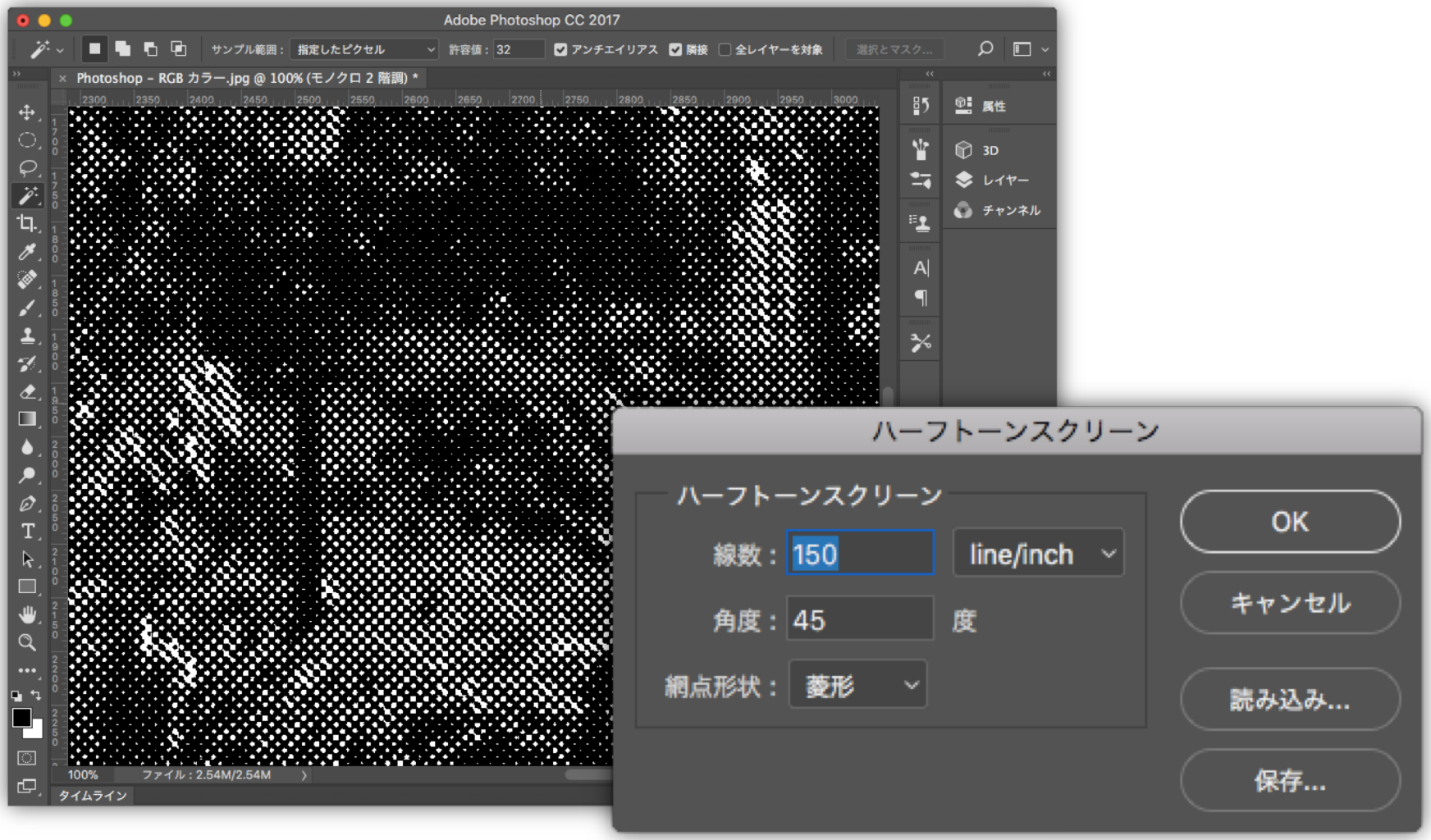
二値画像に変換


モノクロ 2 階調
解像度
入力： 144 pixel/inch
出力： 1200
pixel/inch
OK
キャンセル
種類
使用： ハーフトーンスクリーン...
カスタムパターン：
50% を基準に 2 階調に分ける
パターンディザ
誤差拡散法 (ディザ)
ハーフトーンスクリーン...
カスタムパターン

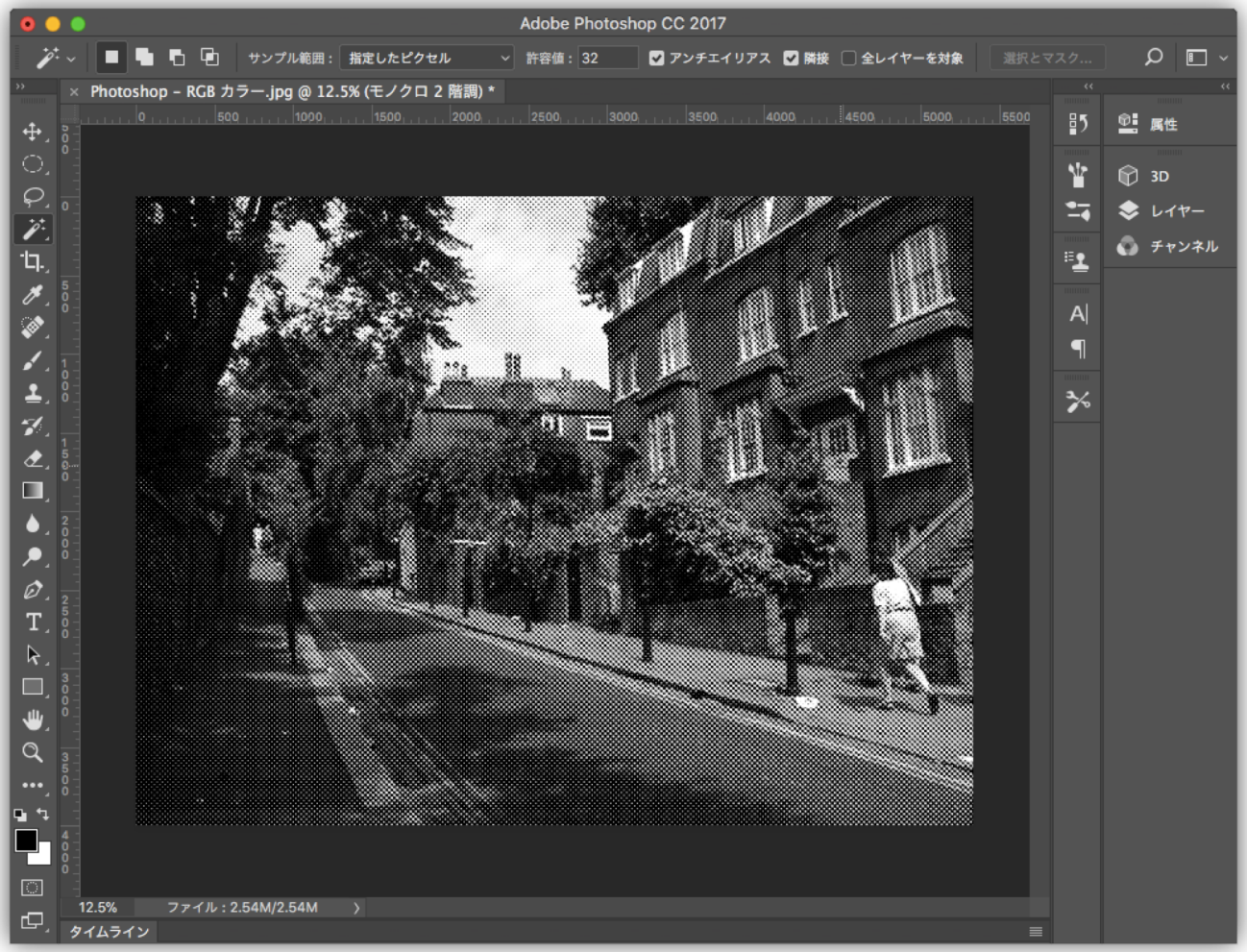
ハーフトーンスクリーン150線



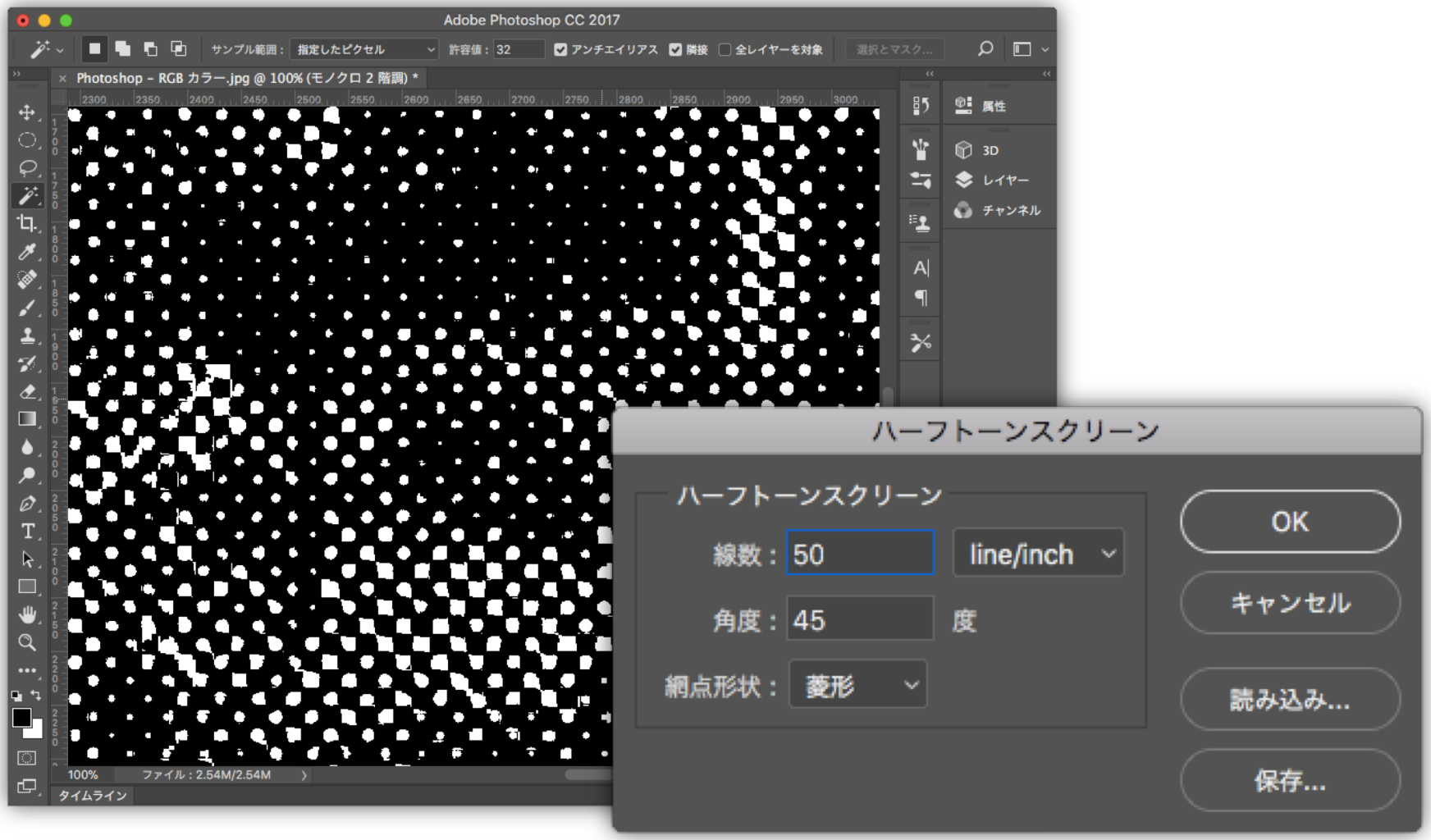
ハーフトーンスクリーン150線 (等倍)



ハーフトーンスクリーン50線



ハーフトーンスクリーン50線 (等倍)

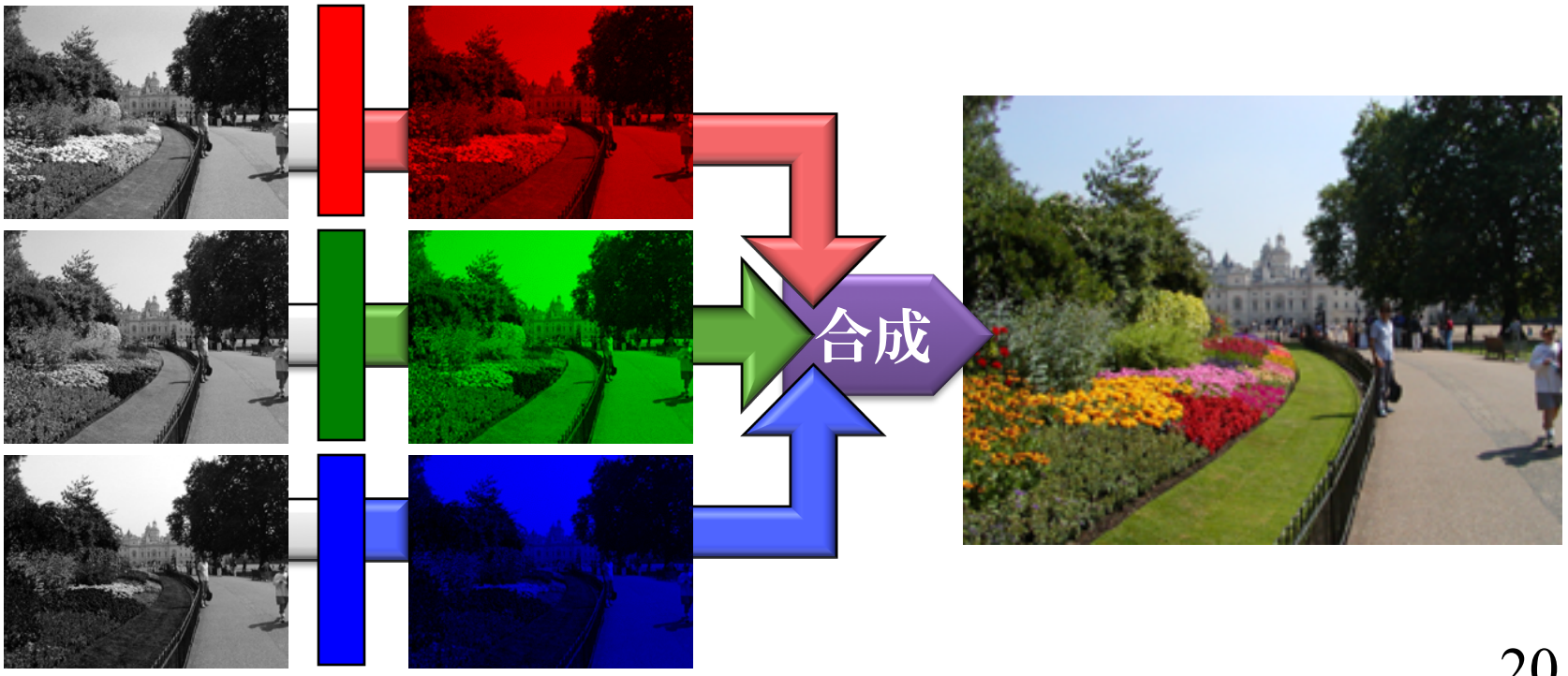


グレースケール画像

- 白黒写真
- 1画素で濃淡を表す
  - 1画素を8bit (1byte)程度で表すことが多い
    - 0→最も暗い
    - 255→最も明るい
  - より多くの階調数(量子化数)を用いる場合もある
    - レントゲン写真(12bit程度)
    - Photoshopの16bitモード
    - 実数で表現する場合

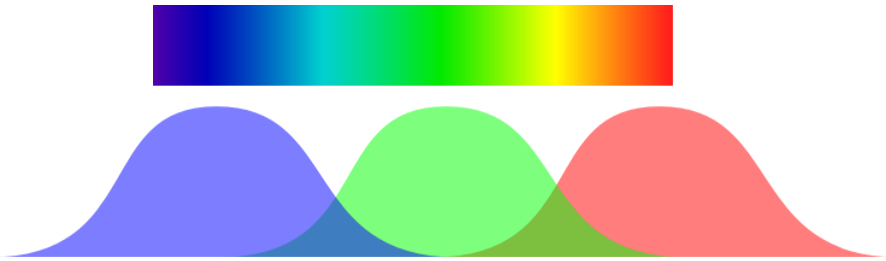
フルカラー画像

- 光の三原色（赤:R，緑:G，青:B）ごとにグレースケール画像で明度を表現する

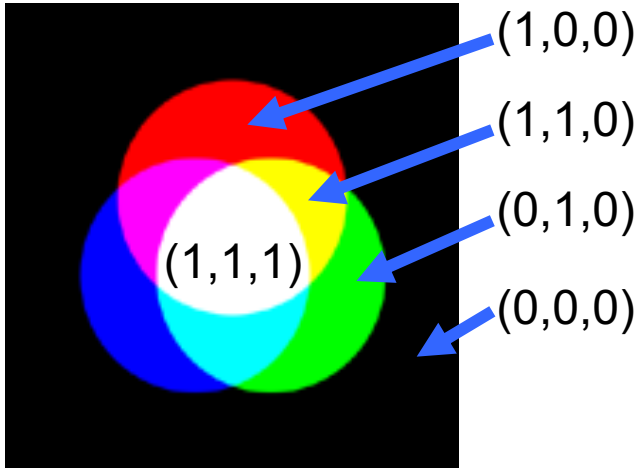


光の三原色

- 人間の視覚は（赤，緑，青）の三色だけ
  - 中間の色は目が「補間」して知覚している
  - 色 C = (R, G, B)



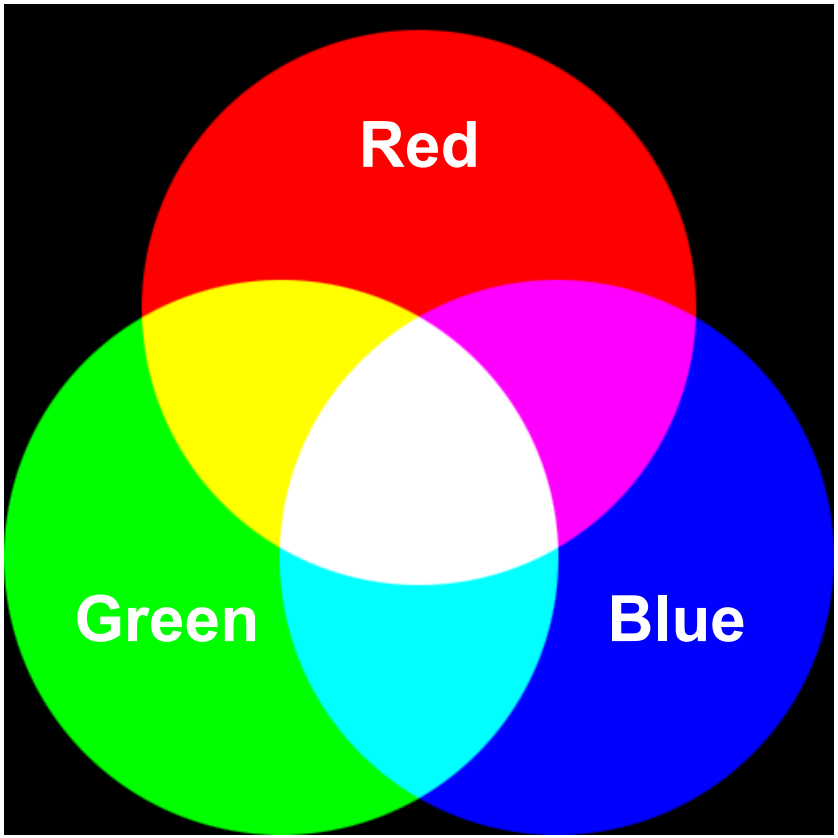
（イメージ図）



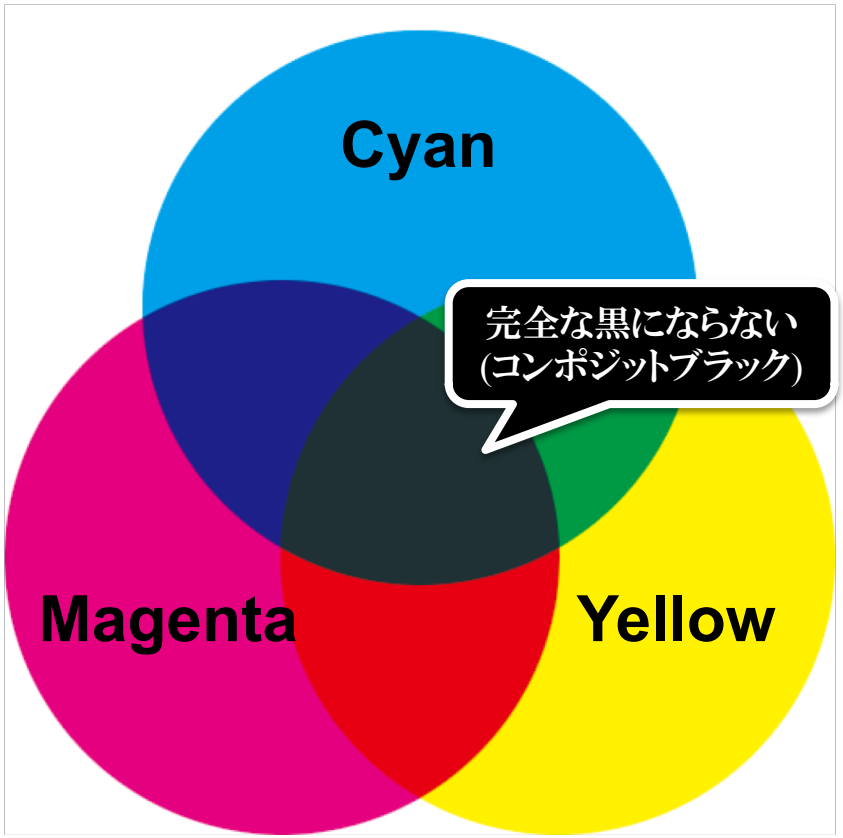
加法混色

加法混色と減法混色

加法混色 (RGB)



減法混色 (CMY)

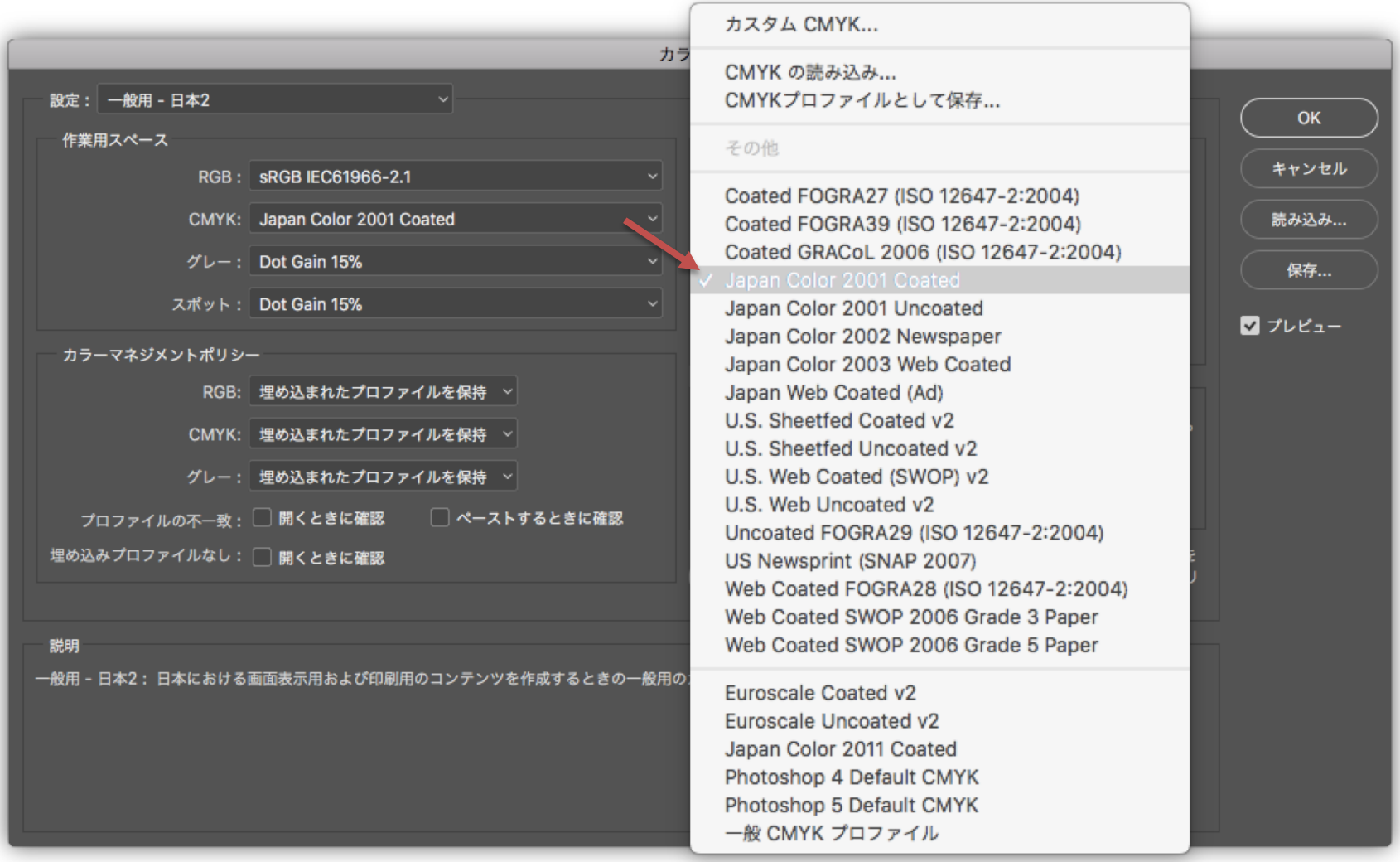


Photoshop のチャンネル

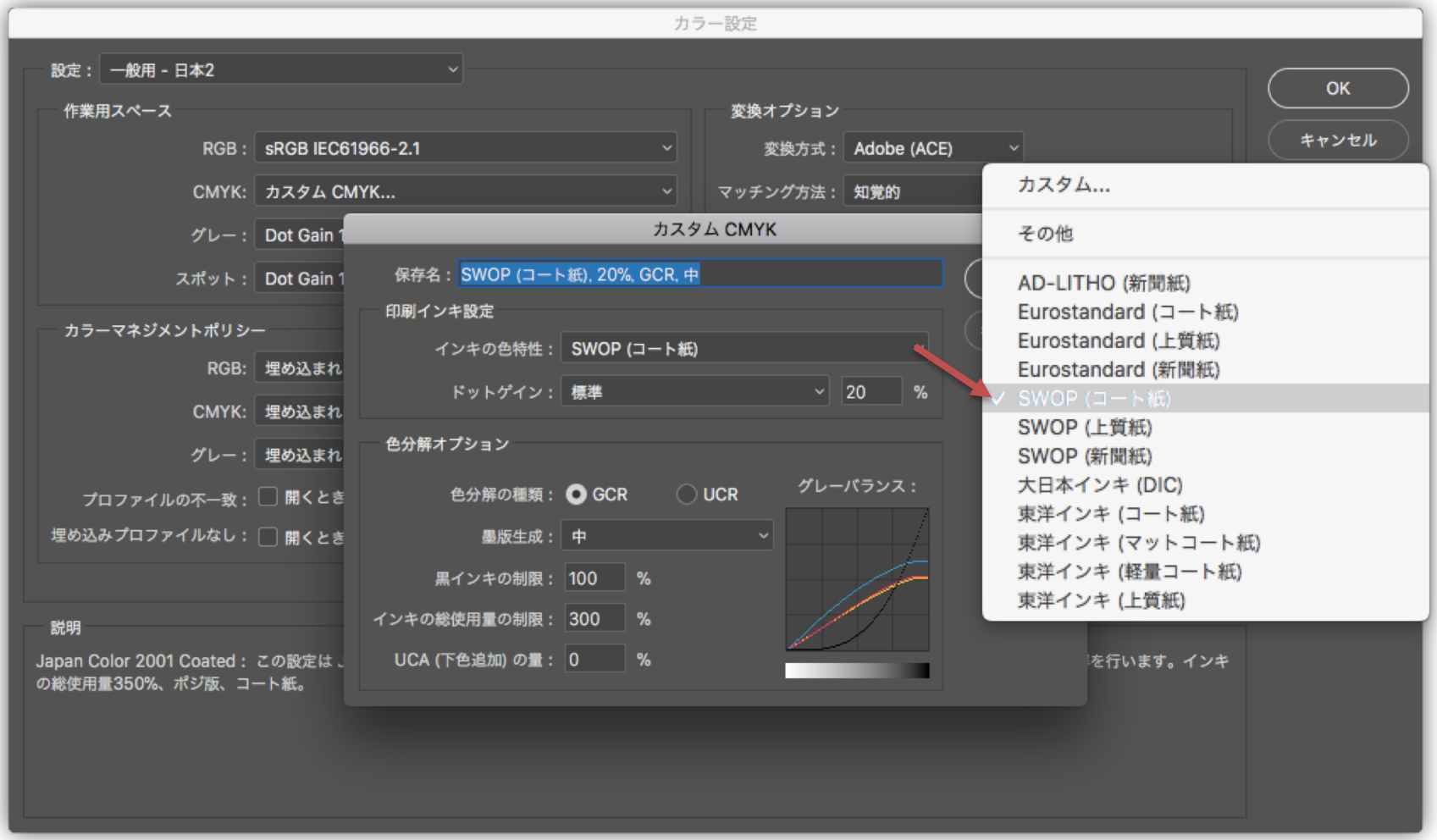
RGB

CMYK

Photoshop のカラー設定



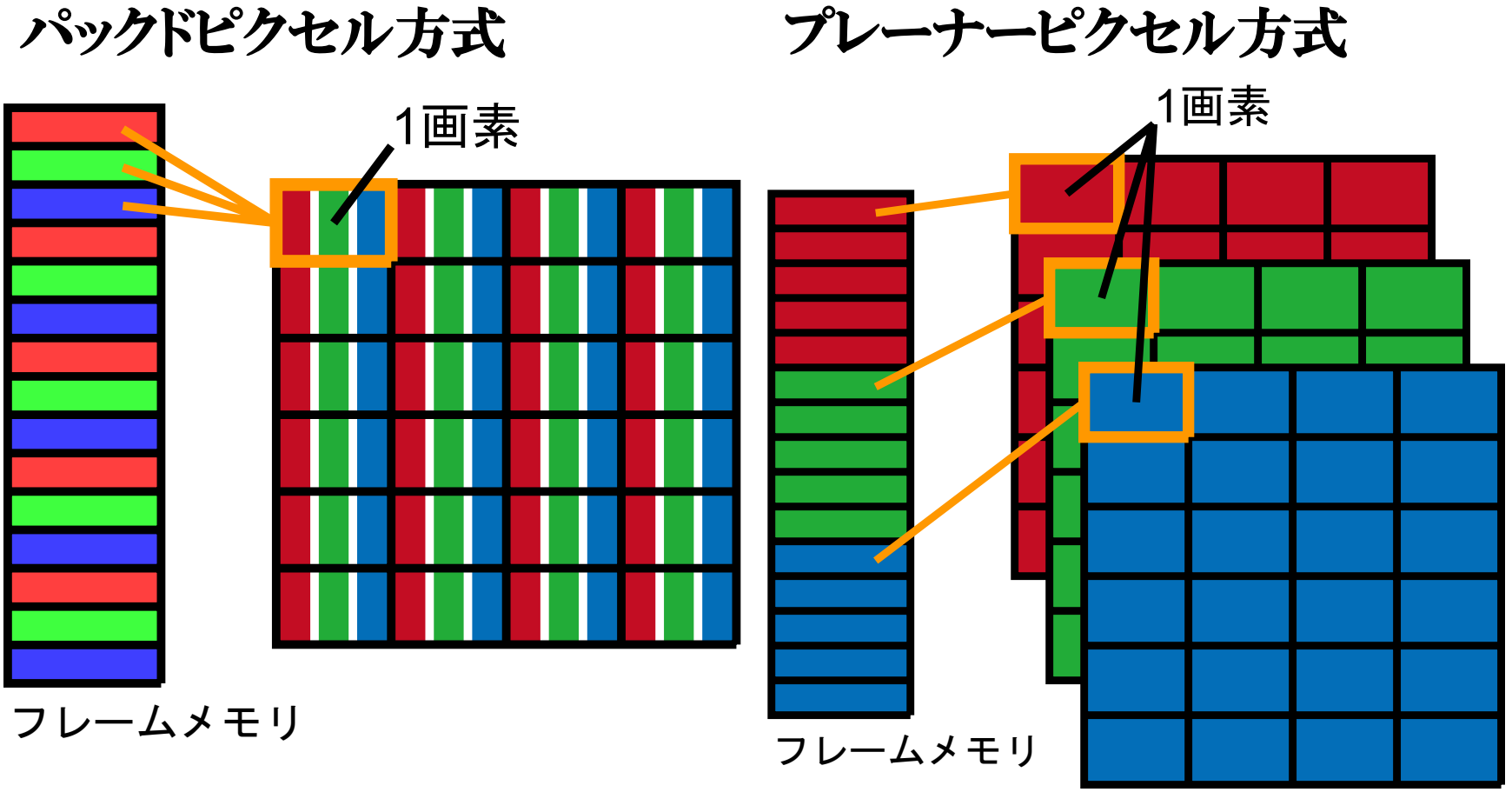
カスタム CMYK


カラー設定
設定：一般用 - 日本2
OK
キャンセル
作業用スペース
RGB：sRGB IEC61966-2.1
CMYK: カスタム CMYK...
グレー：Dot Gain
スポット：Dot Gain
変換オプション
変換方式：Adobe (ACE)
マッチング方法：知覚的
カラーマネジメントポリシー
RGB: 埋め込まれ
CMYK: 埋め込まれ
グレー：埋め込まれ
プロファイルの不一致：開くとき
埋め込みプロファイルなし：開くとき
説明
Japan Color 2001 Coated：この設定は
の総使用量350%、ポジ版、コート紙。
を行います。インキ
カスタム CMYK
保存名：SWOP (コート紙), 20%, GCR, 中
印刷インキ設定
インキの色特性：SWOP (コート紙)
ドットゲイン：標準 20 %
色分解オプション
色分解の種類：GCR UCR
墨版生成：中
黒インキの制限：100 %
インキの総使用量の制限：300 %
UCA (下色追加) の量：0 %
グレーバランス：
カスタム...
その他
AD-LITHO (新聞紙)
Eurostandard (コート紙)
Eurostandard (上質紙)
Eurostandard (新聞紙)
SWOP (コート紙)
SWOP (上質紙)
SWOP (新聞紙)
大日本インキ (DIC)
東洋インキ (コート紙)
東洋インキ (マットコート紙)
東洋インキ (軽量コート紙)
東洋インキ (上質紙)

CMYK はインクの色

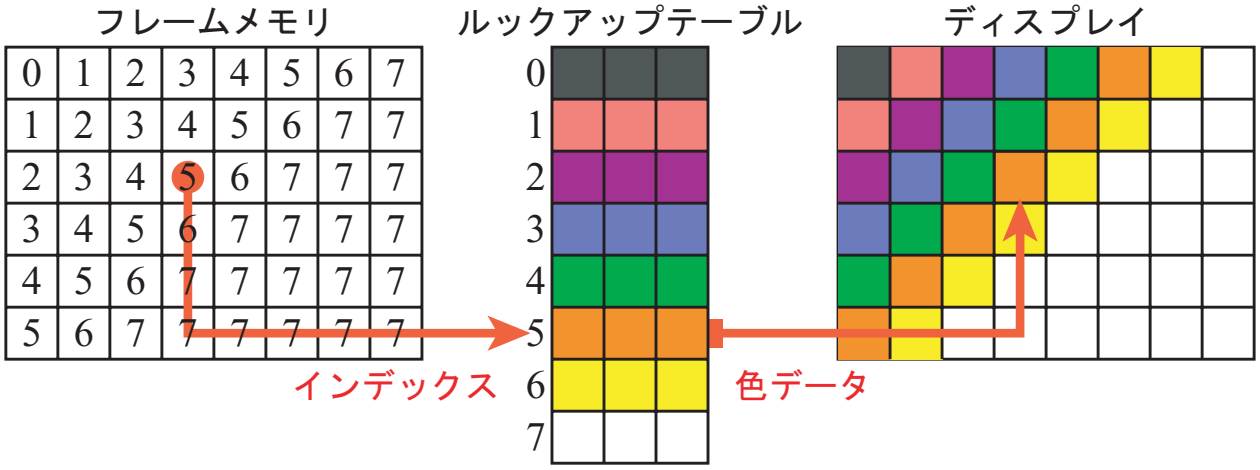
- CMYK で正しい色を出すにはインクの発色を知る必要がある
  - 分版（色分解）には専門的知識が必要
  - 印刷屋さんと相談する
- 素人は sRGB で統一しておくのが無難
  - 普通の PC 用ディスプレイは CMYK を表示できない
  - 家庭用インクジェットプリンタは6色とかあって元から CMYK ではない（ドライバが sRGB から変換）
- 色域の話 http://www.iwashi.org/archives/3342

フルカラー画像のデータ構造



インデックスカラー画像

- フレームメモリにはルックアップテーブルの指標（インデックス）を格納する
  - 色データはルックアップテーブルを参照して得る
  - GIF画像等

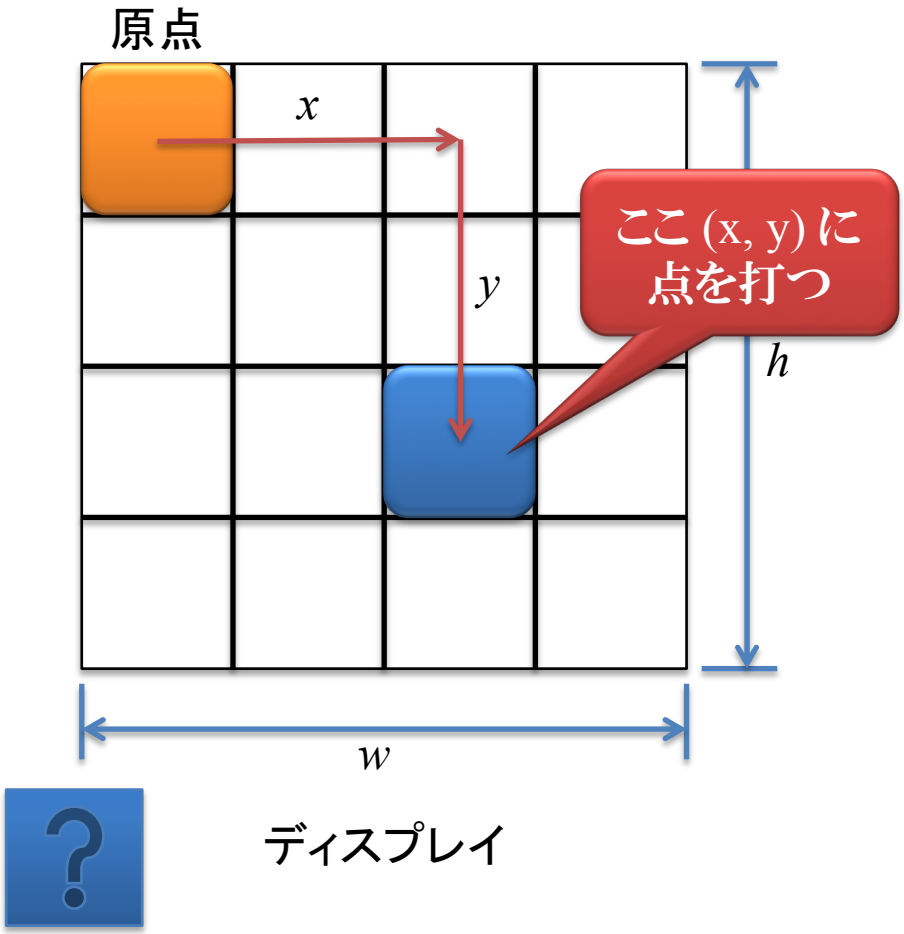


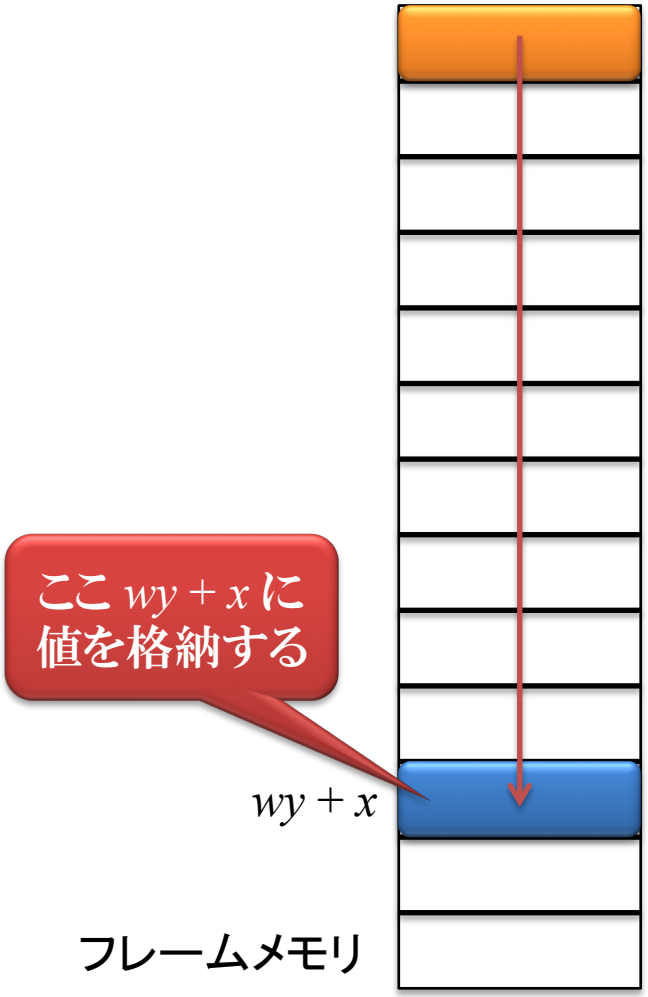
ラスターグラフィックス画像の生成

フレームメモリにデータを格納すれば
画像として再現される

表現したい形に色のデータを格納する

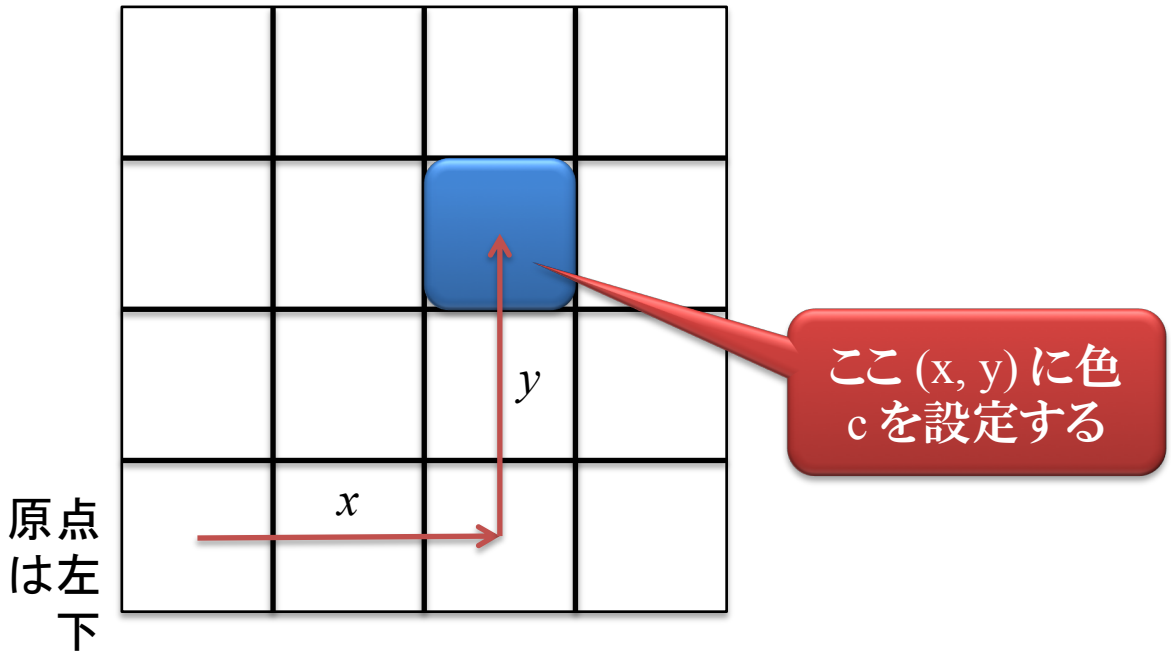
点を描く





点を描く関数 point(x, y, c)

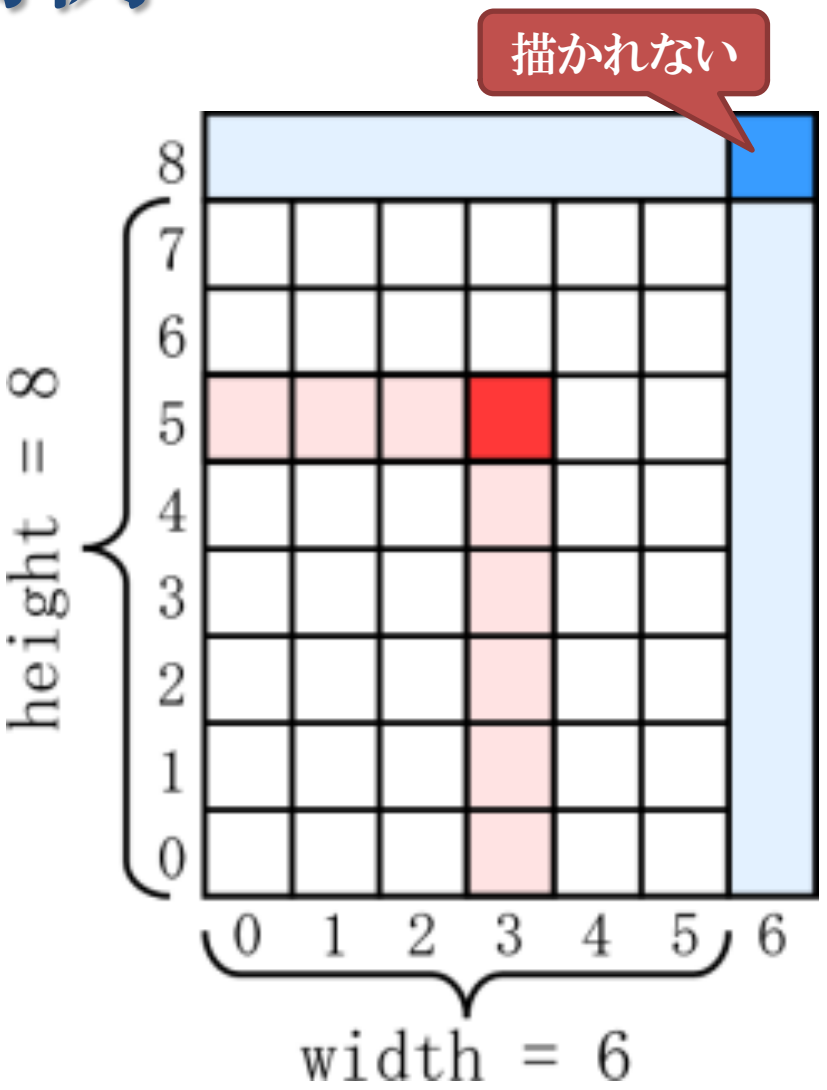
- void point(int x, int y, const double *c)
  - (x, y) の位置の画素に色 c を設定する
  - c[0] : 赤, c[1] : 緑, c[2] : 青, 0 ≦ c[0], c[1], c[2] ≦ 1



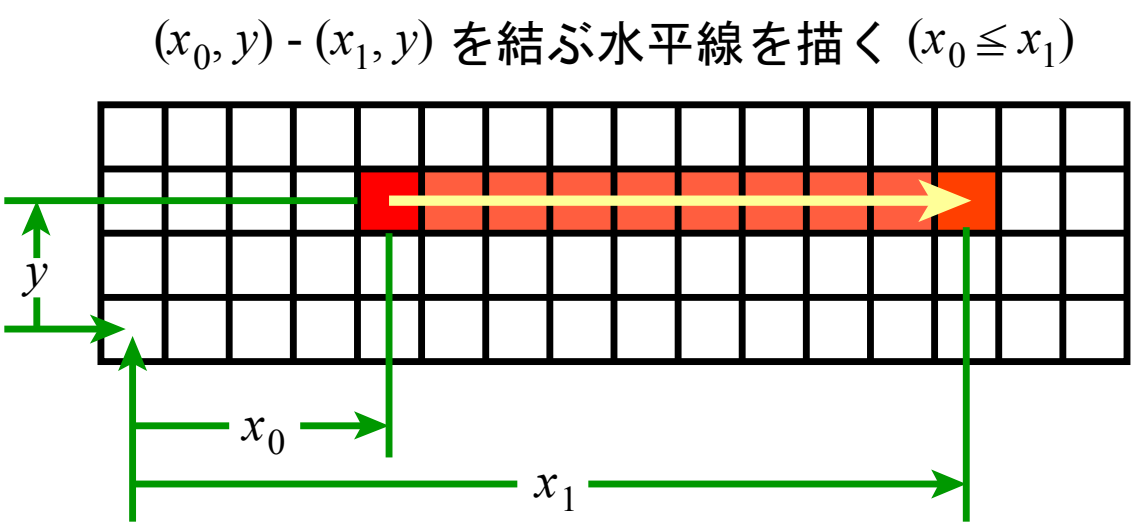
point() 関数の使用例

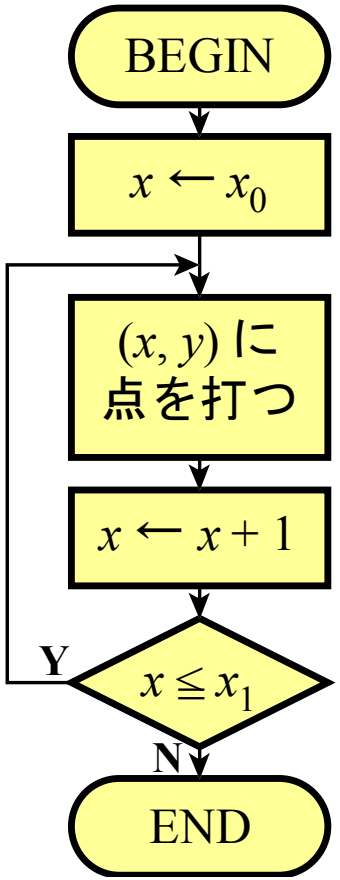
```
/* 点の色 */
const double red[] = {
  1.0, 0.0, 0.0
};
const double blue[] = {
  0.0, 0.0, 1.0
};

/* 図形を描く */
void draw(int width, int height)
{
  point(3, 5, red);
  point(width, height, blue);
}
```

表示領域の幅と高さ

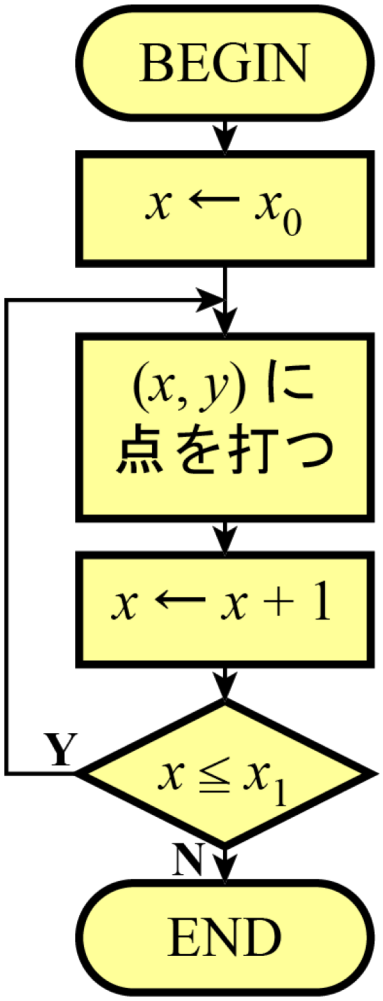


水平線を描く


(x_0, y) - (x_1, y) を結ぶ水平線を描く $(x_0 \leqq x_1)$
y
x_0
x_1


BEGIN
$x \leftarrow x_0$
(x, y) に
点を打つ
$x \leftarrow x + 1$
Y
$x \leqq x_1$
N
END

C 言語に書き直すと



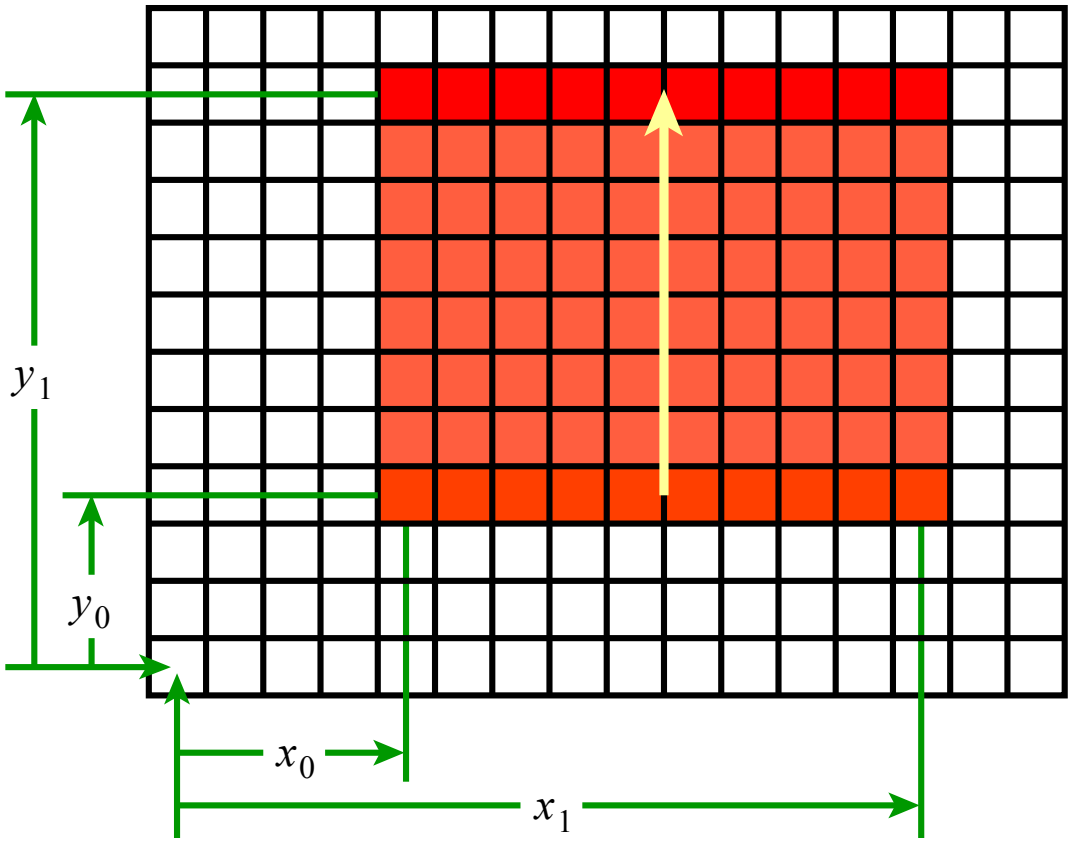

```
/* 水平線を描く */
int x;
x = x0;
do
{
  point(x, y, c);
  ++x;
}
while (x <= x1);
```

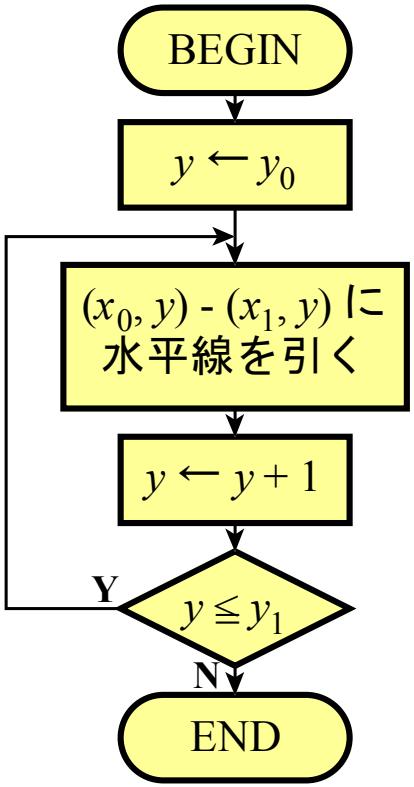
```
/* 水平線を描く */
int x;
for (x = x0; x <= x1; ++x)
{
  point(x, y, c);
}
```

```
/* 水平線を描く */
int x;
x = x0;
while (x <= x1)
{
  point(x, y, c);
  ++x;
}
```

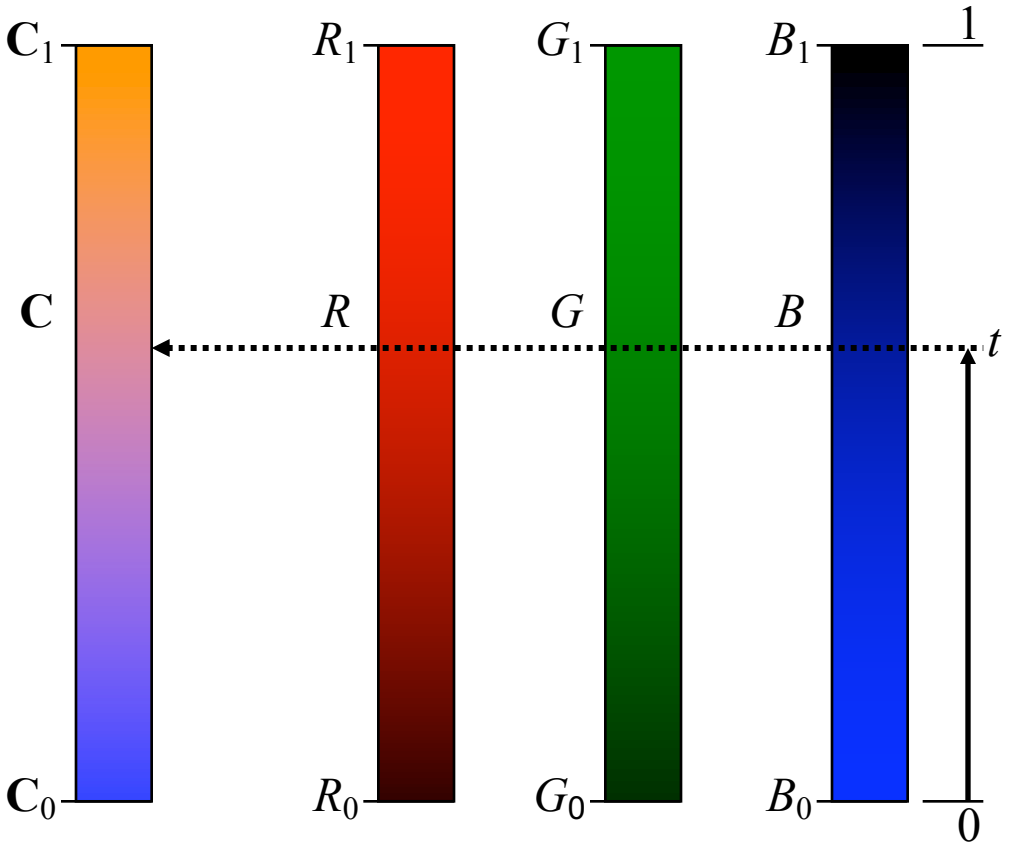
領域を塗りつぶす

(x_0, y_0) - (x_1, y_1) を対角線とする
矩形領域を塗りつぶす $(x_0 < x_1, y_0 < y_1)$





グラデーション

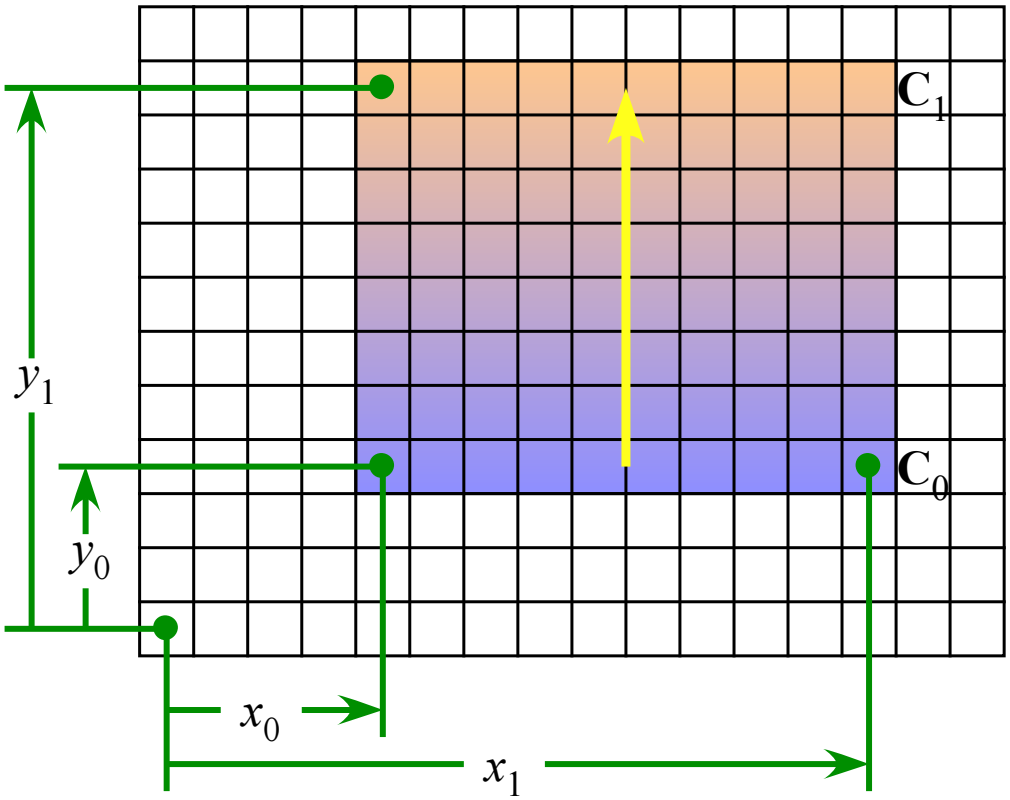


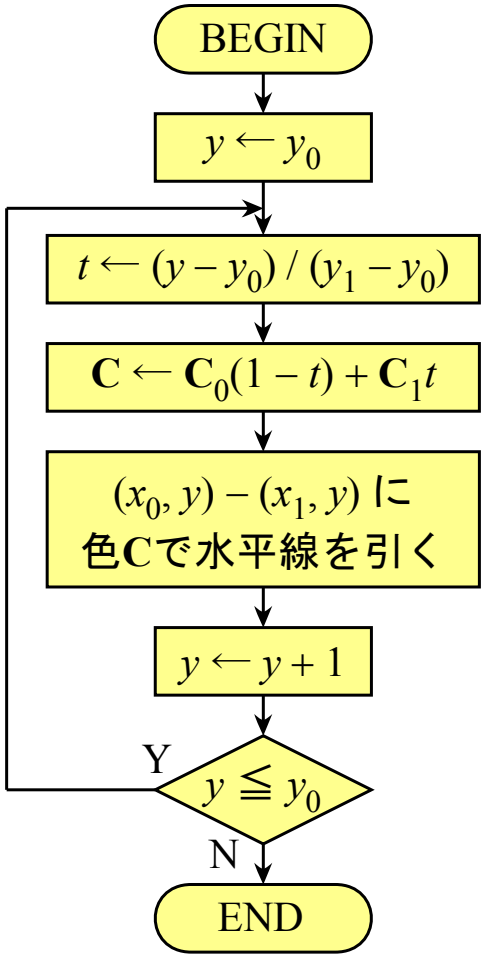
線形補間の場合

$$\mathbf{C} = \mathbf{C}_0(1-t) + \mathbf{C}_1 t$$

$$R = R_0(1-t) + R_1 t$$
$$G = G_0(1-t) + G_1 t$$
$$B = B_0(1-t) + B_1 t$$

グラデーションで塗りつぶす

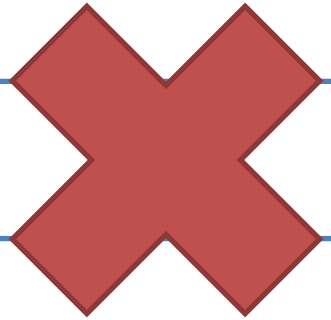




データ型に注意

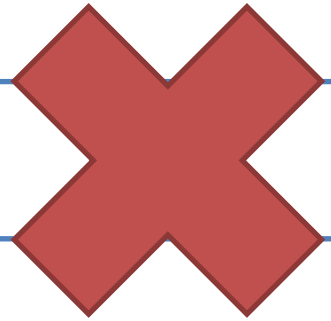
x, y, x0, y0 は int（整数）型

```
int t;
t = (y - y0) / (x - x0);
```



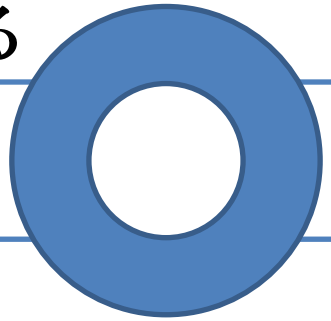
t を double（倍精度実数）型にする

```
double t;
t = (y - y0) / (x - x0);
```



右辺を double 型にキャスト（型変換）する

```
double t;
t = (double)(y - y0) / (double)(x - x0);
```



データ型

整数型

- char
- unsigned char
- short
- unsigned short
- int
- unsigned
- unsigned int
- long
- unsigned long

整数定数

- `'A'`
- `65`
- `0x41`
- `-100`
- `1L`

実数型

- float
- double

実数定数

- `0.75`
- `.0075e2`
- `0.075e1`
- `.075e1`
- `75e-2`

おわり