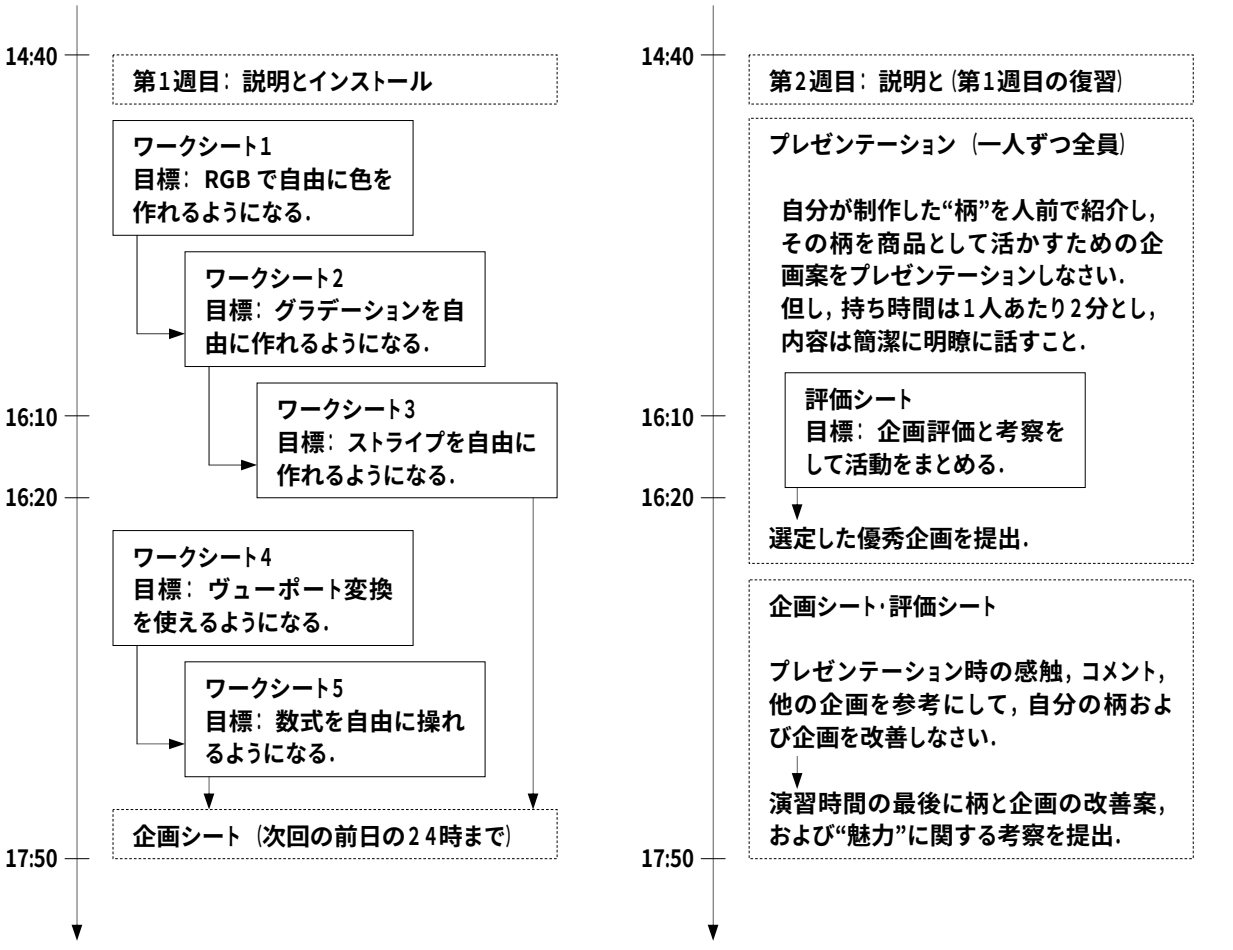
数式がつくるかたち

概要：どのような CG 画像も，基本は描画領域を構成するすべての画素（ピクセル）に対して何らかの方法で決定された色をあてはめる（発光させる）作業によって作成されている．平面を単色で塗りつぶす場合も陰影や光沢のある立体表現も画素の色の配列にすぎない．色の決め方は，直接数値で指定する方法から複雑な計算式を用いて決定する方法までさまざまである．本演習ではプログラミングをベースとした画像生成の基礎として，各画素の色の決定を数式あるいは単純なルールを用いて行ない，2 次元パターンを作成する．「数式」や「プログラミング」と聞くと難しそうな印象をもつかもしれないが，これまでに学んだ知識で十分対応できる内容であり，むしろコンピュータがつくり出すいくつものパターンを楽しむことができるはずである．
キーワード：ディジタル画像，画素，RGB，グラデーション，ストライプ，ヴューポート変換，剰余パターン．

第1週目はまず，CG における色彩の基礎として，RGB の数値の指定方法と対応する色を確認し，グラデーション，ストライプパターンについて学ぶ．次に，色々な数式が生成する剰余パターンについて学び，演習内容を駆使した独自の柄およびその活用企画を考案する．

第2週目は，一人ずつ全員に，出来上がった柄およびその活用企画のプレゼンテーションをしてもらう．また，プレゼンテーションされた柄および活用企画の評価は全員（学生自身）におこなってもらい，教員の評価と合わせて成績評価に反映させる．



評価は，1週目の出席点，2週目のプレゼンテーション点（出席点），課題の提出点，努力点，評価点の総合成績とする．なお，評価点は学生自身による評価結果も加味するものとする．また，課題1の提出期限は2週目の前日までとする．

片倉コレクション――季節ごとに絵を着替えよう！

●片倉コレクション

あなたは“片倉コレクション”という企画をおこなう会社のスタッフだとしましょう．いま，あなたの会社では，この商品にはこの柄，あるいはそのバリエーション，そして季節ごとにどんなバリエーションを用意するか，そういった“柄”とそれをどう活用するかに関する企画の立案･収集をおこなっています．

●数式がつくるかたち――柄を企画しよう！

“数式がつくるかたち”の演習では，(1)コンピュータ･グラフィックスを用いてどんな“柄”をつくることができるのかの演習，そして(2)そこでできた“柄”あるいはできそうな“柄”をどのような商品にテクスチャ･マッピング（貼り付け）するかの企画演習，この2点をおこなっていただきます．

●人を説得する――プレゼンテーション入門

この演習で立案した企画および“柄”をもとに，顧客，そして代理店に対してプレゼンテーションをおこない，売り上げを伸ばす必要があります．そのためにはどんなプレゼンテーションをするのが効果的かに関しては，後期の“人を説得する”というテーマでおこないます．勿論，この演習で提出したあと，改良を重ねて，後期に発表していただいても結構です．

●WWW-データベースシステム入門

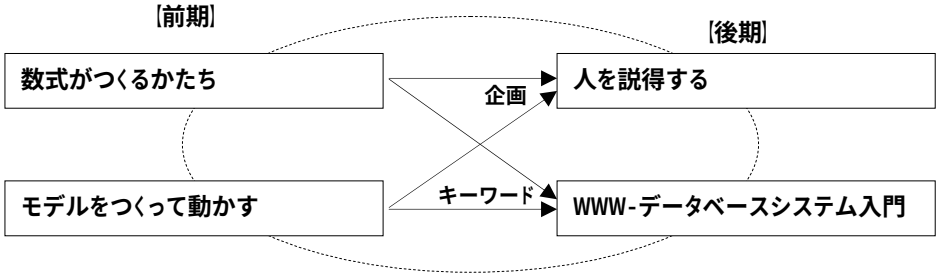
あなたは“片倉コレクション”のスタッフですから，スタッフ全員がつくった企画，および柄をデータベースとして管理し，注文に対しては迅速に対応する必要があります．そのためのデータベースをどのようにつくったらよいかに関する演習は，後期の“WWW-データベースシステム入門”でおこないます．

●モデルをつくって動かす――万博を企画しよう！

なお，前期の“モデルをつくって動かす”のテーマでは，同じく“片倉コレクション”のスタッフとして，万博やテーマパークの企画，および企画したもののモデルをつくる，ということをおこない，その内容は同じく，後期の“人を説得する”および“WWW-データベースシステム入門”で続きをおこないます．

●その他――自分の作品を大切にしよう！

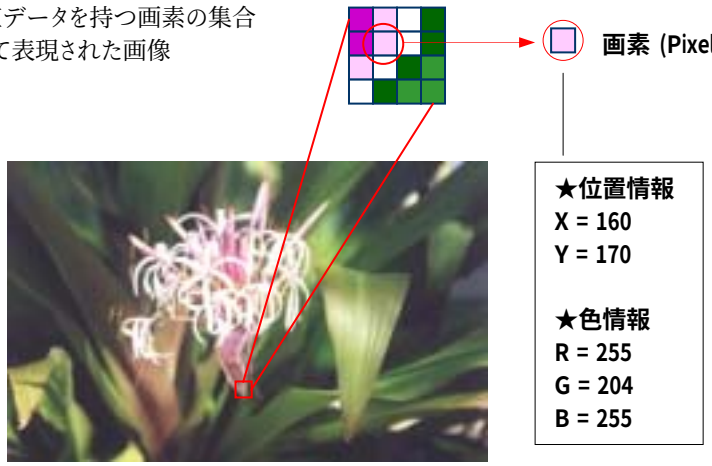
他のテーマで学んだことも，是非，どのように活用することができるか，について考えてみてください．直接的には商品とは結びつきにくい内容であっても他の技術を支える重要な役割を担っていることもあります．例えば，“UNIX”や“コンピュータネットワーク”は，“WWW-データベース”を支えています．各テーマで学んだ内容を何に活かせるかを考え，作品や企画は大切にして改良を重ね，就職活動に役立ててください．



「片倉コレクション」――季節ごとに絵を着替えよう！

●ディジタル画像

数値データを持つ画素の集合として表現された画像



■次のものはディジタル画像か？

- 家に飾ってある絵
- その絵を写真で撮ったそのフィルム
- その絵の写真のプリント
- スキャナーでコンピュータに取り込んだ絵
- ディスプレイに表示された絵
- プリンタで印刷された絵
- コンピュータ･グラフィックスでつくった絵

■“数式がつくるかたち”でおこなうこと

位置情報 (X, Y) ⇒ 色情報 (R, G, B)

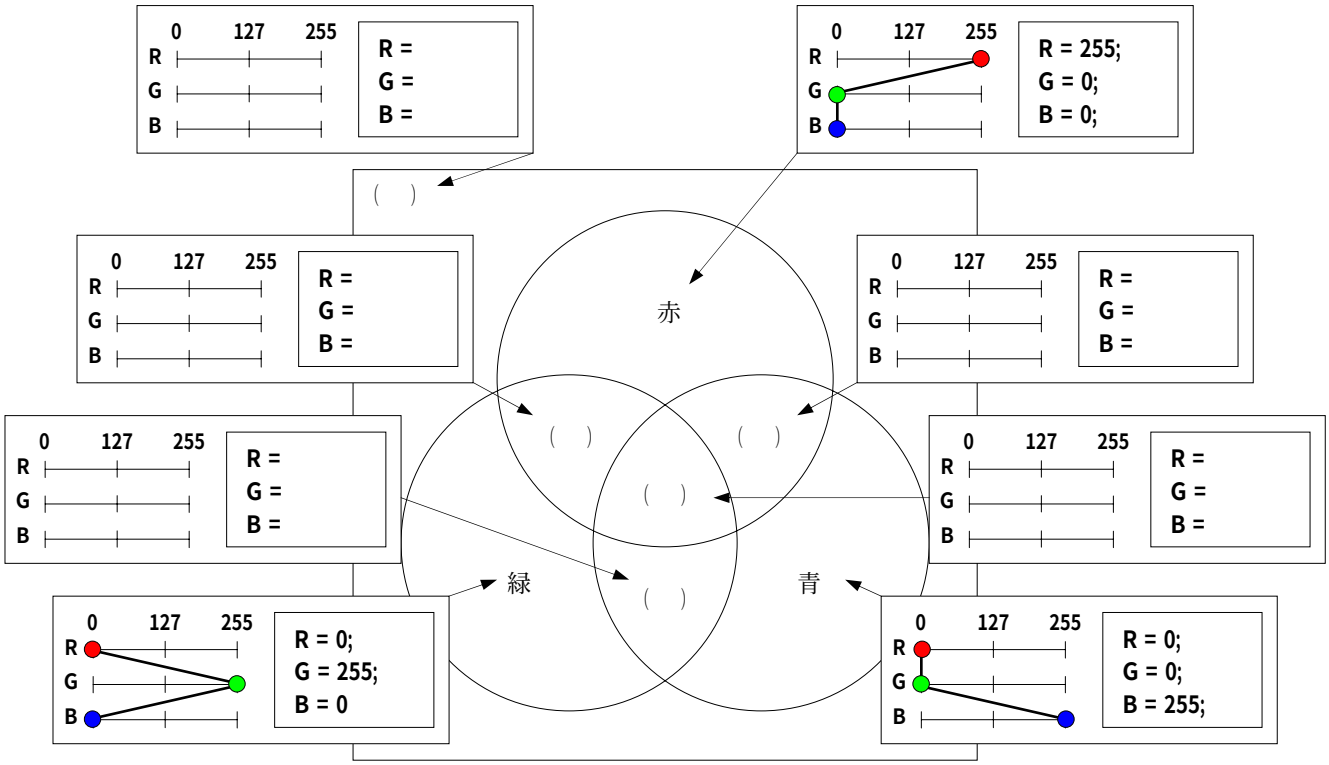
つまり，位置情報から色情報を算出する規則を数式によって表現しているのである．

ワークシート１： 色を作ってみよう

目標: RGB で自由に色を作れるようになる.

色彩表現

1 画素（ピクセル）あたりどれだけのメモリが割り当てられるかによって，モノクロ画像であれば濃淡の階調がきまり，カラー画像であれば扱える色数が決まる．たとえば，各画素に 8 ビットのメモリが割り当てられているとすると，$2^8 = 256$ となり，256 段階の濃淡や 256 種類の色を使うことができる．色光の 3 原色は赤･緑･青(RGB)で，CG の場合，この 3 色の混合でほとんどの色をつくることができる．絵の具では，混ぜる色数が多くなるほど濁って暗くなるが，色光の場合は明るくなり 3 原色を混合すると白になる．前者の場合を**減法混色**，後者を**加法混色**という．R, G, B それぞれに 8 ビット与えられている場合，256 階調（0～255 の値をとる）であるから，理論的に可能な色数は $(2^8)^3 = 16{,}777{,}216$ となる．この色数は人間の識別能力に対して十分に対応した数であるとされている．

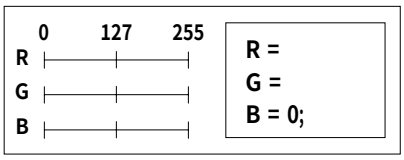


作業手順

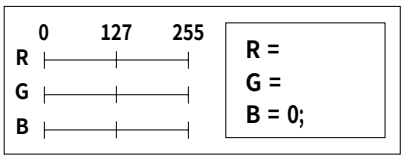
1. 上図は加法混色の様子を表したものです．まず，括弧内に色名を記入してください（5 箇所）.
2. 赤･緑･青の部分には RGB の数値と折れ線グラフが既に記入してあります．プログラム 1 の所定の箇所にその数値を記入して実行し，その色が表示されることを確認してください.
3. この例を参考にして，残りの 5 色の数値とグラフを記入してください．また，必ずプログラムに数値を記入して，正しく色ができていることを確認してください.
4. **挑戦してみよう**をやってみてください.
5. （中級者向け） RGB で複数の色を混ぜるとき，数値演算にはどのような法則性が見られるのかを観察し，RGB で色を混ぜるときの公式を考案してください.

挑戦してみよう

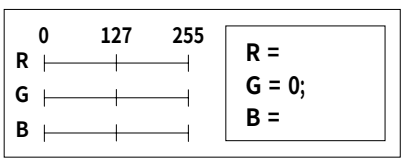
橙（オレンジ）色



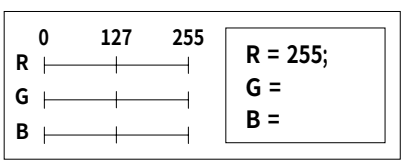
黄緑色



青紫色



桃（ピンク）色

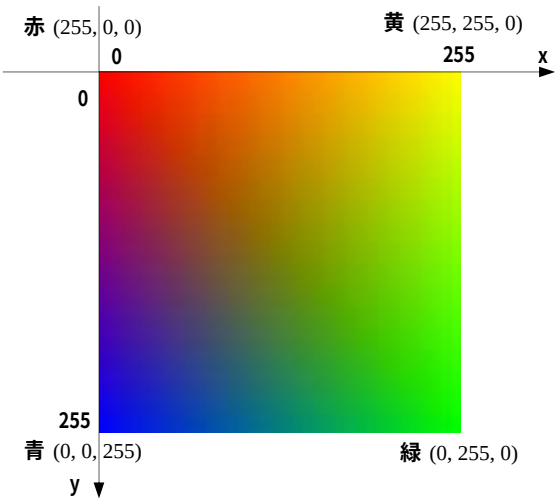


ワークシート２： グラデーションを作ってみよう

目標: **グラデーションを自由に作れるようになる.**

座標系
この演習では, ディスプレイの左上を原点とする**スクリーン座標系**を基本とし, **y** 軸は下方に向かって正となるものとする.

グラデーション(gradation)
グラデーションとは, 2 色の間を少しずつ変化させて, 滑らかに一方の色から他方の色に移行させることである.

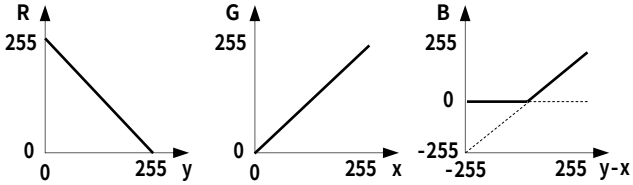


グラデーションの作り方

1. 図の四隅に, 色とその RGB の数値を記入する.
2. RGB ごとに四隅の数値を集めて変化の規則性を観察する.

$$R=\begin{pmatrix}255 & 255\\ 0 & 0\end{pmatrix} \quad G=\begin{pmatrix}0 & 255\\ 0 & 255\end{pmatrix} \quad B=\begin{pmatrix}0 & 0\\ 255 & 0\end{pmatrix}$$

3. 変化の方向 (**x** のみ, **y** のみ, **x** と **y**) を確認したらグラフを描く.



4. グラフを参照しながらプログラムに記す数式を決定する.

```
R = 255 - y;
G = x;
B = y - x;
```

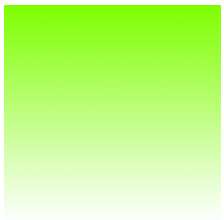
5. 実際にプログラムを実行して, 正しく表示されることを確認する.

作ってみよう（プログラム1の所定の箇所に記入）

1. 左端がオレンジ色, 右端が白色, となるグラデーションを作ってください.

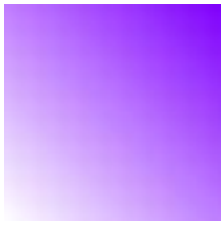
```
R = 255;
G =
B =
```

2. 上端が黄緑色, 下端が白色, となるグラデーションを作ってください.



```
R =
G = 255;
B =
```

3.（中級者向け）左下端が白色, 右上端が青紫色, となるグラデーションを作ってください.



```
R =
G =
B = 255;
```

ワークシート３： ストライプを作ってみよう

目標: ストライプを自由に作れるようになる.

位置情報によって色情報を制御する
画素は位置情報(x, y)と色情報(R, G, B)から構成されており，前者で後者を制御する方法として次にストライプを演習する.

ストライプ(stripe)
ストライプとは，複数（通常は 2 つ）の異なる色領域を一定の規則に基づいて繰り返し表示させることである.

```
// *** BEGIN YOUR PROGRAM *** //

if( (x / 16) % 2 == 0 ){
  R = 191 - (x - y)/4;
  G =  63 - (x - y)/4;
  B = 127 + (x - y)/2;
}
else{
  R = 255 - y/2;
  G = 255;
  B = 255 - y;
}

// *** END YOUR PROGRAM *** //
```

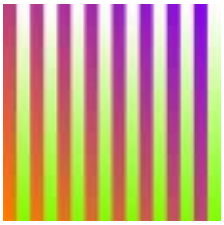
（色領域 1）

（色領域 2）

ストライプを作るためには，この条件式の挙動を理解しておく必要がある. まず，x は整数であるため，x が 0〜15 の時に 16 で割ると 0 になり，これを 2 で割った余りは 0 であるから，この時には**色領域 1** が描画されることになる. 同様に，x が 16〜31 の時に 16 で割ると 1 になり，これを 2 で割った余りは 1 であるから，この時には**色領域 2** が描画されることになる. つまり，16 ごとに 2 つが交互に描画されるのである.

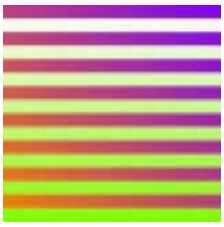
x	(x / 16)	(x / 16) % 2	色領域
0〜15			
16〜31			
32〜47			
48〜63			

作ってみよう（プログラム1に記入）



1. 縦縞模様

条件式:
(x / 16) % 2 == 0

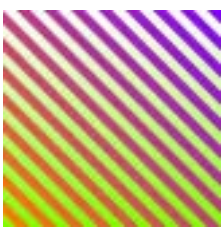


2. 横縞模様

条件式:
(/ 16) % 2 == 0

3.（中級者向け）斜縞模様

条件式:
(/ 16) % 2 == 0



4.（中級者向け）斜縞模様

条件式:
(/ 16) % 2 == 0

挑戦してみよう

5.（上級者向け）横波縞模様

条件式:

6.（上級者向け）斜波縞模様

条件式:

7.（上級者向け）同心円模様

条件式:

8.（上級者向け）螺旋縞模様

条件式:

ワークシート４： ウィンドウの大きさを変えてみよう

目標: **ヴューポート変換を使えるようになる.**

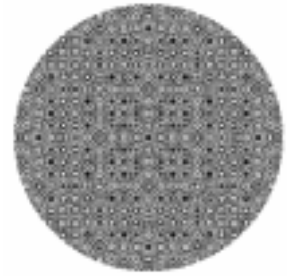
ヴューポート変換

ヴューポート変換の詳細は, このテキストの課題2のプログラムの横に記した説明を参照してください. ここでは, ウィンドウ（画面に表示する数式の領域のこと）を変化させると表示のされ方がどのように変化するのかを探求してください.

作業手順

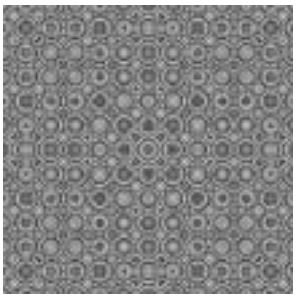
1. 表示されている図形を作るためには, 指定された数式に, どのようなウィンドウを設定したらよいかを探ってください.
2. （中級者向け） 原点**(0, 0)**を中心としない非対称な領域を設定し, 図形の変化の具合を探ってください.
3. （上級者向け） 縦横の比率を変えた領域をを設定し, 図形の歪み具合を探ってください.

作ってみよう 2 乗の剰余（プログラム2に記入）



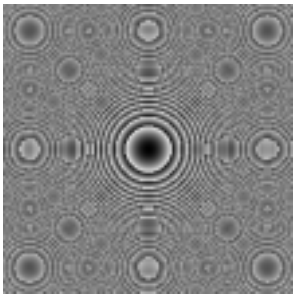
数式: $X^2 + Y^2$

ウィンドウ:
sX =
sY =
wX =



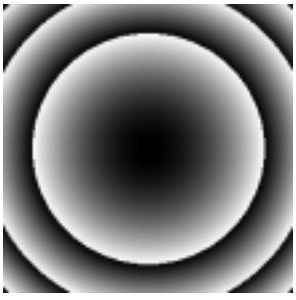
数式: $X^2 + Y^2$

ウィンドウ:
sX = -1000;
sY = -1000;
wX = 2000;



数式: $X^2 + Y^2$

ウィンドウ:
sX = -100;
sY = -100;
wX = 200;



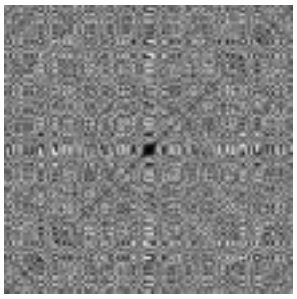
数式: $X^2 + Y^2$

ウィンドウ:
sX =
sY =
wX =

作ってみよう 3 乗の剰余

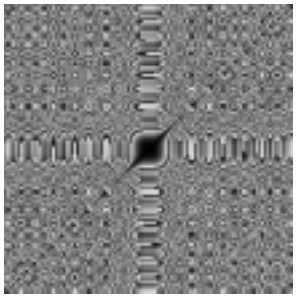
数式: $\left|X^3 + Y^3\right|$

ウィンドウ:
sX =
sY =
wX =



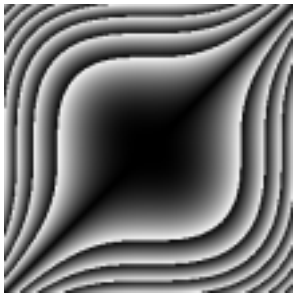
数式: $\left|X^3 + Y^3\right|$

ウィンドウ:
sX = -100;
sY = -100;
wX = 200;



数式: $\left|X^3 + Y^3\right|$

ウィンドウ:
sX = -50;
sY = -50;
wX = 100;

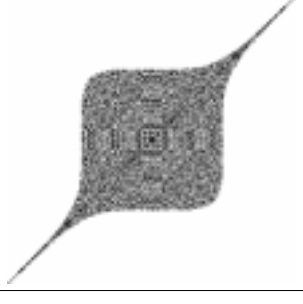


数式: $\left|X^3 + Y^3\right|$

ウィンドウ:
sX =
sY =
wX =

数式: $X^4 + Y^4$

ウィンドウ:
sX = -250;
sY = -250;
wX = 500;



数式: $\left|X^5 + Y^5\right|$

ウィンドウ:
sX = -150;
sY = -150;
wX = 300;

ワークシート５： いろいろな数式を使ってみよう

目標: **数式を自由に操れるようになる.**

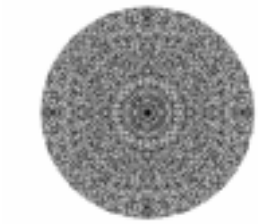
数式がつくるかたち

多くの場合, 予期しないパターンが現れるが, 単純な数式が複雑なパターンをつくりだす面白さを味わうことができる.

作業手順

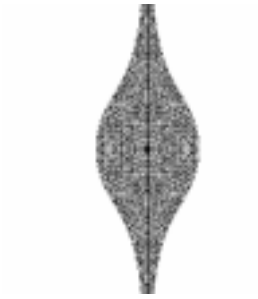
1. 表示されている図形をつくるためには, どのような数式とその領域(ウィンドウ)を設定したらよいか探ってください.
2. 自分で独自の数式を作り, またその領域を変化させ, 自分の気に入った図形を見つけてください.

作ってみよう 多項式の剰余（プログラム2に記入）



数式: $\left|X^2+Y^2\right|^{2.5}$

ウィンドウ:
sX =
sY =
wX =



数式: $\left|X^3+XY^2\right|^2$

ウィンドウ:
sX =
sY =
wX =

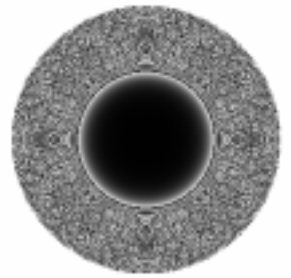
数式: $\left|\frac{(X/7)^4}{Y}-Y^2\right|$

ウィンドウ:
sX = -70000;
sY = -70000;
wX = 140000;

数式: $\left|X^2-Y^2\right|^5$

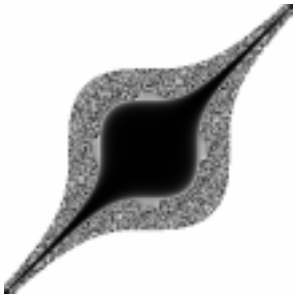
ウィンドウ:
sX = -100;
sY = -100;
wX = 200;

挑戦してみよう（中級者向け） **指数関数･三角関数の剰余**



数式: $e^{\left|X^2+Y^2\right|}$

ウィンドウ:
sX = -5;
sY = -5;
wX = 10;

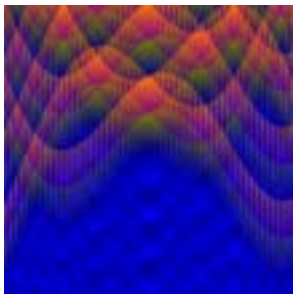


数式:

ウィンドウ:
sX = -5;
sY = -5;
wX = 10;

数式:

ウィンドウ:
sX = -5;
sY = -5;
wX = 10;



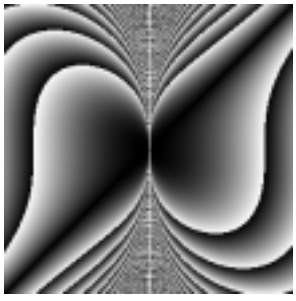
数式: $X(\cos X)-Y$

ウィンドウ:
sX = -500;
sY = -500;
wX = 1000;

挑戦してみよう（上級者向け） **多項式の剰余･相似図形の発見**

数式:

ウィンドウ:
sX = -70000;
sY = -70000;
wX = 140000;



数式:

ウィンドウ:
sX = -20;
sY = -20;
wX = 40;

企画シート： 柄を活かす企画を考えよう

課題1 独自の柄，およびそれを活かす企画（1次案）

グラデーション，ストライプ，剰余パターンを駆使し，線と線の間隔，線の幅，配色も考えて独自の“柄”をつくりなさい．また，その柄を商品として活かすための企画案を考えなさい．

1. ソースコードは **プログラム2** を使用すること．
2. 演習した技法は積極的に取り入れること．
3. ヴューポートのサイズは **480*480** とすること．
4. 以下のものを所定の方法で提出すること．
   ソースコード：**u02p000_2.java**
   画像ファイル：**u02p000_2.bmp**
   企画（下記の内容を 200～400 字程度でまとめる）

企画
★その柄の“印象”を表す“形容詞”を5つ記してください．

★その柄の“印象”は．．．

ネガティブな ＋＋＋＋＋＋＋ ポジティブな

★その柄を使用するのに適切な“季節”は？

春　夏　秋　冬

具体的には？

★その商品が使用される“時代背景”は？

★ターゲットとする“顧客層”は？

★その柄に適切な“タイトル”をつけてください．

★その柄を使用するのはどのような商品ですか？

★その他，企画にあたって考えたことを記してください．

課題2 企画内容の改善（2次案）

発表時の聴衆の反応，および寄せられたコメントを参考にして，“柄”および企画を改善しなさい．

1. 積極的に柄および企画の改善をおこなうこと．
2. 何を改善したのか？　どう改善したのか？
3. その改善により本当に良くなったのか？
4. 改善後の柄および企画を所定の方法で提出すること．
   ソースコード：**u02p000_2.java**
   画像ファイル：**u02p000_2.bmp**
   企画（下記の内容を 200～400 字程度でまとめる）

企画の改善案（発表時のコメント等も記しておく）
★改善した点を左側と対照しながら記してください．

評価シート： まとめ

目標: 柄と企画の評価と考察をして制作活動をまとめる.

評価のポイント

1) 印象を表す形容詞は適切か？
2) 印象の評価(ポジティブ―ネガティブ)は適切か？
3) 想定している季節は適切か？
4) 時代背景はどうか？
5) ターゲットとしている顧客層はどうか？
6) 柄のタイトルはどうか？
7) その商品の総合的な魅力はどうか？

作業手順

1. 柄と企画, およびプレゼンテーションの内容を評価する.
   a. 発表者の学籍番号と氏名を記す.(自分を含む)
   b.「評価のポイント」の7項目に対して:
      ◎＝大変良い, ○＝良い, △＝イマイチ, ×＝論外
   の4段階で評価する.
2. その中から優秀企画を3つ選び報告する.(自分も対象)
3. “魅力”ある商品を企画するためには何が大切か？
   に関して考察をし, 自分の制作活動をまとめる.

番号	氏　名	形容	評価	季節	時代	顧客	題名	魅力	番号	氏　名	形容	評価	季節	時代	顧客	題名	魅力

考察 “魅力”ある商品を企画するためには何が大切か？

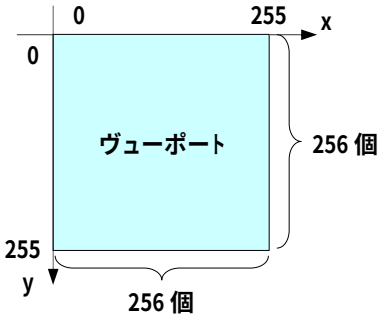
プログラム１のソースコード（学習用）

```
//
// u02p000_1.java for TMLib
//
import jp.ac.teu.media.*;

public class u02p000_1 extends TMLib {
  final static int vsize = 256;

  public static void main(String [] args) {
    begin(new u02p000_1(), vsize, vsize);
  }

  public void Main() {
    int x, y, R, G, B;
    TMLibImage  img = new TMLibImage();
    img.newImage(vsize, vsize);

    for(x = 0; x < vsize; x++){
      for(y = 0; y < vsize; y++){

        // *** BEGIN YOUR PROGRAM *** //

        R = 255;
        G = 255;
        B = 255;

        // *** END YOUR PROGRAM *** //

        img.setRGB(x, y, R, G, B);
      }
    }
    drawImage(0, 0, img);
    img.writeImage("u02p000_1.bmp");
  }
}
```

この演習では「TMLib」というグラフィックス用のライブラリを使用します．受講生は予め以下の URL を参照して TMLib を正しくインストールしておいてください．
[URL] http://www.teu.ac.jp/media/~earth/TMLib/

ファイル名の **u02p000_1.java** の網掛の部分，及びプログラム中の4箇所の網掛で示した部分は全て自分の学籍番号に置き換えてください．（プログラム2も同様にして書き換え）

「**vsize**」は **view port size** を表しています．ここでは「**256**」と指定しているので縦横 256 ピクセルの正方形がヴューポートとして設定されます．
この値は変更しないでください．なお，ピクセル数は 256 だが，座標は **0〜255** までの値となるので注意してください．



この演習で編集する対象となるのはこの部分です．その他は特に編集する必要はありません．また，テキストの本文にはこの部分のみを掲載してあります．

作成した画像は，プログラムを実行し，画面に画像が表示されたときに，自動的に保存されるようになっています．従って受講生は改めて画像を保存する必要はありません．また，メニューの「File」→「Save」は別の形式で画像が保存されるため，絶対に使用しないでください．

ソースコードは，以下の URL から取得できるので，予習をする方は利用してください．なお，受講生には，学籍番号などを記入したものを演習時間内に配布する予定です．
[URL] http://www.teu.ac.jp/media/~tagiru/math/

数式で使える数学関数とその使用例

絶対値：**Math.abs(double a)**
　色情報にマイナスの値を指定することができないため，数式の計算結果がマイナスになった場合には絶対値を使用してプラスの数値に変換する必要がある．
　例：**Z1 = Math.abs(X*X - Y*Y);**

指数関数：**Math.exp(double n), Math.pow(double a, double n)**
　自然底数(**Math.E**)の **n** 乗，**a** の **n** 乗を計算する．
　例：**Z1 = Math.exp(X*X + Y*Y);**
　　Z2 = Math.pow(X*X + Y*Y, 2.5);

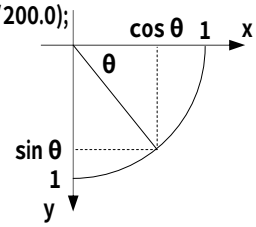
対数関数：**Math.log(double a)**
　自然底数(**Math.E**)を底とする対数を計算する．
　例：**Z1 = Math.log(X*Y);**

平方根：**Math.sqrt(double a)**
　例：**Z1 = Math.sqrt(X*Y);**

三角関数：**Math.sin(double a), Math.cos(double a), Math.tan(double a)**
　弧度法（180 度が π (**Math.PI**)という数値で表現される）によって与えられた角度に対する，正弦，余弦，正接を計算する．正弦と余弦の値は **-1〜1** で変化する．
　例：**Z1 = 100.0 * Math.sin(X*Math.PI/180.0) + 127.0;**

三角逆関数：**Math.asin(double a), Math.acos(double a), Math.atan(double a)**
　正弦，余弦，正接の逆計算をする．
　例：**Z1 = 80.0 * Math.asin(X/200.0);**

注意：数式を作る際には，変数が変化した時に，**0** で割り算をするという事態が発生しないように気をつけてください．



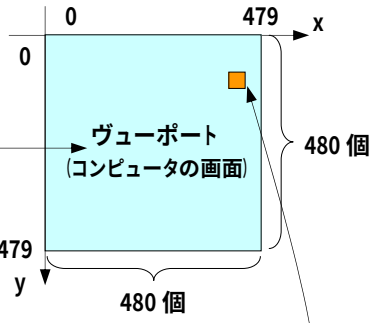
プログラム２のソースコード（提出用）

```
//
// u02p000_2.java for TMLib
//
import jp.ac.teu.media.*;

public class u02p000_2 extends TMLib {
  final static int vsize = 480;

  public static void main(String [] args) {
    begin(new u02p000_2(), vsize, vsize);
  }

  public void Main() {
    int x, y, R, G, B;
    double X, sX, wX, dX, Y, sY, wY, dY, Z1, Z2, Z3;
    TMLibImage  img = new TMLibImage();
    img.newImage(vsize, vsize);

    // *** BEGIN WINDOW *** //

    sX = -100.0;
    sY = -100.0;
    wX = 200.0;
    wY = wX;

    // *** END WINDOW *** //

    dX = wX/(double)vsize;
    dY = wY/(double)vsize;

    for(x = 0; x < vsize; x++){
      X = sX + dX * (double)x;

      for(y = 0; y < vsize; y++){
        Y = sY + dY * (double)y;

        // *** BEGIN YOUR PROGRAM *** //

        Z1 = X*X + Y*Y;
        Z2 = 0;
        Z3 = 0;
        R = (int)Z1 % 256;
        G = (int)Z1 % 256;
        B = (int)Z1 % 256;

        // *** END YOUR PROGRAM *** //

        img.setRGB(x, y, R, G, B);
      }
    }

    drawImage(0, 0, img);
    img.writeImage("u02p000_2.bmp");
  }
}
```

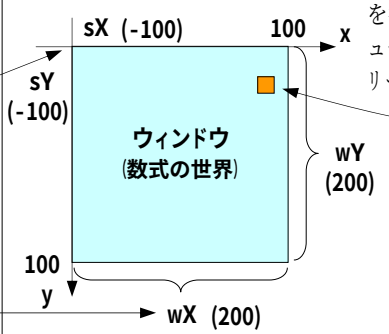
ヴューポート変換

スクリーン座標系の指定領域に，対象となる画像を表示するための処理をヴューポート変換という．

ディスプレイ画面上の指定された矩形領域を**ヴューポート**といい，原図形のうち表示したい部分を指定する矩形領域を**ウィンドウ**という．



ヴューポート変換を行うため，コンピュータの画面（スクリーン座標系）における 1 ピクセルが，数式の世界（ワールド座標系）のどんな値に対応するのかを計算しておく必要がある．



例：dX = wX / vsize = 200 / 480 = 0.4167

コンピュータの画面（スクリーン座標系）の座標**(x)**に対応する数式の世界（ワールド座標系）の座標**(X)**は，始点**(sX)**に，1 ピクセルあたりの長さ**(dX)**と座標値**(x)**を掛けたものを足し合わせることによって計算することができる（**Y** も同様）．

例：**X = sX + dX * x = -100 + 0.4167 * 100 = 141.67**

数式の適用

実際に使用したい数式を用いて計算をしている．ここでは，例として「$X^2 + Y^2$」という数式を適用している．

例：$Z1 = X^2 + Y^2 = (141.67)^2 + (141.67)^2 = 40140.776$

なお，括弧書きの変数名「**(double)**」はキャスティング（型変換）というもので，もともと整数型**(int)**で定義された「**vsize**」や「**x**」や「**y**」を臨時に実数型**(double)**として扱うためのもので，音楽でいうところの臨時記号のようなものである．コンピュータでは型の違う変数同士の演算は直接行えないため，このようにして型を合わせてから計算をする必要がある．

剰余パターン

計算結果を色情報に変換する計算をしている．ここでは，例として「**R = (int)Z1 % 256**」という数式を適用している．まず，実数の変数 Z1 を整数に型変換している．この段階で小数点以下は四捨五入されて「**Z1 = 40141**」と値が多少変化する．これを丸め誤差という．次に，256 で割った余りを計算する．この値は 0～255 の範囲を繰り返すパターンを形成する．

例：**R = Z1 % 256 = 40141 % 256 = 205**

補足説明

CG による画像生成

コンピュータが扱う図形や画像の情報は数値化されたディジタルデータであり，そのデータが，対応するコンピュータディスプレイ上に図形や画像となって表示される．数値データをもつ画素とその集合であるこのような画像をディジタル画像という（ディジタル画像の詳細は，本テキスト「ディジタル画像処理の基礎」を参照）．2 次元の図形を作成するためのソフトウェアは数多く市販されており，目的に応じて線や面の情報をそれらの関係で扱うドロー系や画素単位で扱うペイント系などを使って容易に画像の作成や変換を行なうことが出来る．しかし，本演習ではこのような既成のソフトウェアは用いないで，ディジタル画像の最小単位である画素の色を決定するプログラムを考えながら，画像作成の基礎を学ぶ．作業は描画領域の画素の位置情報（座標値）を使って何らかのルールや数式に当てはめ，割り出された数値を色の値にするといった単純なものだが，生成される画像は既成のソフトウェアでは作りにくい，あるいは不可能なパターンとなる．

座標と座標系

2 次元平面上の点の位置は直交座標系（**xy** 座標系）の **x** 値と **y** 値で表し，3 次元の場合は **x** 軸および **y** 軸に直交するもう 1 本の軸である **z** 軸が加わる．CG の場合，扱う図形の特質や表現の条件によって，さまざまな座標系が用いられている．特に 3 次元図形の場合は個々の立体固有の座標系であるローカル座標系，それらを配置するワールド座標系，さらに 3 次元データを 2 次元データに投影変換する視野参照座標系，ディスプレイに表示するときのスクリーン座標系などがある．2 次元図形を対象とする本演習では，ディスプレイの左上を原点とするスクリーン座標系を基本とし，**y** 軸は下方に向かって正となるものとする．

数式がつくるかたち

画像の作成には何らかのかたちで数学的処理がなされるため，数式はつきものだが，ここでは数式は直接かたちを作り出すための道具として利用する例を体験する．方程式と聞くと直線や曲線が描かれたグラフを思い出すが，これは式の値を座標値として平面上に表現したものであり，数式がつくるかたちのひとつである．ここでは式の値を座標値ではなく色情報として用い平面を彩色してみる．課題 2 のプログラムは $Z1 = X^2 + Y^2$ を用いて作成したものだが，画面中央を中心とした同心円が波紋のように広がり，周囲にはいくつもの小さな波紋が見られる．このように何らかのプロセスで求めた値を色に置き換えて表示する手法は，フラクタル・パターンとして知られるマンデルブロ集合やジュリア集合を可視化する際に用いられている．数式がつくりだすパターンは，数式を適用する座標系（ワールド座標系）上にあり，スクリーン座標系とは一致していない場合が多い．課題 2 のプログラムのパターンも図の中心が原点**(0, 0)**であり，画面の左上を原点とするスクリーン座標系とは一致していない．この場合，表示したい画像をスクリーン座標系に変換する必要がある．

アルゴリズム（課題 2 のプログラムの場合）

X: 描画領域の **X** 座標値（変数）
Y: 描画領域の **Y** 座標値（変数）
Z1: 計算結果

1. 数式を決める．ここでは $Z1 = X^2 + Y^2$ を使用．
2. 数式を適用する平面を設定する．課題 2 のプログラムの場合は網掛け表示した矩形で，**(sX, sY)**を始点とし，**(-100, -100)**が**(X, Y)**それぞれの初期値になる．数式に座標値を代入すると，$Z1 = (-100)^2 + (-100)^2 = 20000$ となる．
3. **Z1** の値を色データとするが，**0〜255** の範囲におさめるため **256** で除算し，その剰余を色データとする．この剰余値が画素**(X, Y)**の色になる．
4. ビューポート内の全ての画素について 2 および 3 の処理を繰り返す．

剰余を用いているため，**0〜255** までの数値が規則的に繰り返される．したがって生成されるパターンも規則性を帯びたものになる．

ヒント：同心円模様・螺旋縞模様を作るには以下の二つのメソッドが必要となります．
（「**public void Main()**」の直前の行に追加）

```
// 2 点間の長さを求めるメソッド（追加）
public double getLength(int x, int y, int ox, int oy){
  return Math.sqrt((double)((x - ox)*(x - ox) + (y - oy)*(y - oy)));
}

// 中心点(ox, oy)に対する角度（ラジアン）を求めるメソッド（追加）
public double getRadian(int x, int y, int ox, int oy){
  double r = getLength(x, y, ox, oy);
  double sth = Math.asin((double)(y - oy)/r);
  double cth = Math.acos((double)(x - ox)/r);
  if( 0 <= sth && sth <= Math.PI/2.0 && 0 <= cth && cth <= Math.PI/2.0 ) return sth;
  else if( 0 <= sth && sth <= Math.PI/2.0 && Math.PI/2.0 <= cth && cth <= Math.PI ) return cth;
  else if( -1.0*Math.PI/2.0 <= sth && sth <= 0 && Math.PI/2.0 <= cth && cth <= Math.PI ) return Math.PI - sth;
  else if( -1.0*Math.PI/2.0 <= sth && sth <= 0 && 0 <= cth && cth <= Math.PI/2.0 ) return 2.0*Math.PI - cth;
  else return 0.0;
}
```

追加