

2019年度コンピュータグラフィックス 中間小テスト

場所 : 3201 教室
(3号館2階, 屋上にテニスコート)
日付 : 5月31日(金)
時間 : 14:40から15~20分程度

C言語ソースファイルの穴埋め問題
持込み・参照は一切不可

受験には学生証が必要！ 座席指定！ 遅刻に注意！

小テスト終了後にOD2教室に移動して通常の授業をします。

授業のおおまかな予定

4/05	ガイダンス	2次元グラフィックス 四角形, 円形, 直線の描画 カラー画像 座標変換
4/12		
4/19		
4/26		
5/10		
5/17	休講	中間試験(小テスト)

5/24 休講
5/27(月) 6限 補講(OD2)

5/31 (3201教室集合) ← 中間試験(小テスト)

6/07	3次元グラフィックス 物体形状の表現 シェーディング 隠面消去 OpenGL CG作品	期末試験
6/14		
6/21		
6/28		
7/01(月) 6限 補講(OD2)		
7/05	休講	試験期間中
7/12		
7/19		

試験期間中 ← 期末試験

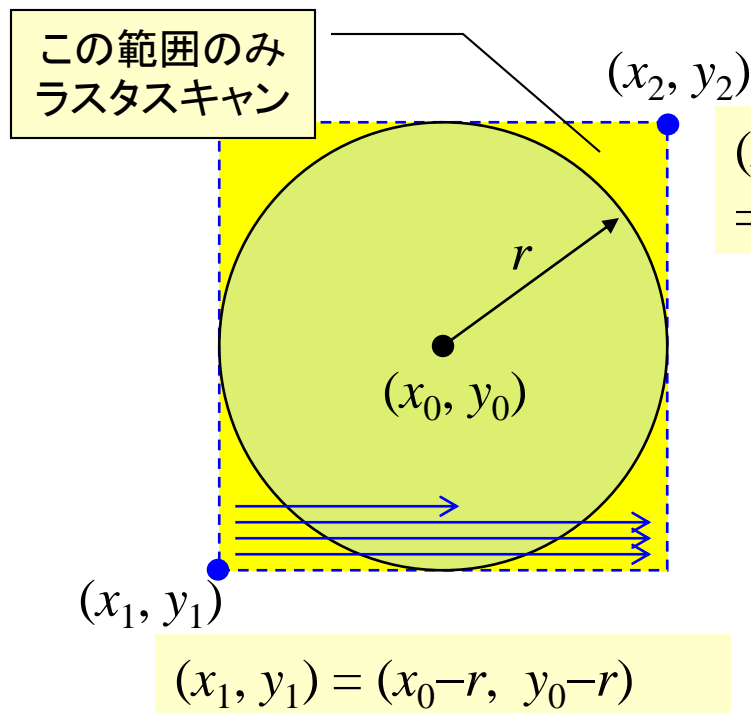
レポート	10%
出席	10%
修了作品	20%
中間試験	20%
期末試験	40%

レポート
≡ C言語プログラム + α

受講するためには
→ 「基礎プログラミング」程度
のC言語の知識が必須

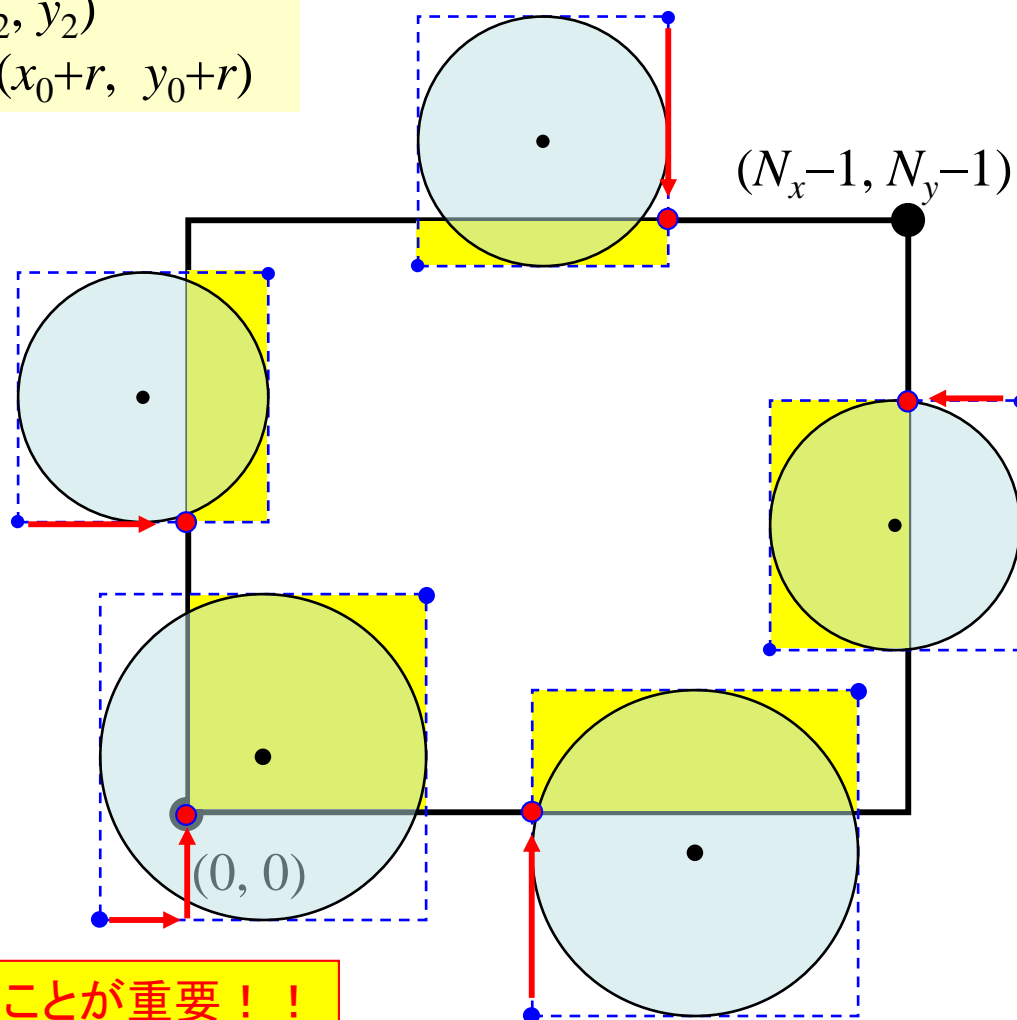
円塗りつぶし関数の考え方

クリッピング
= 描画を必要な領域だけに限定すること



$x_1 < 0$ なら,	$x_1 = 0$
$x_2 > N_x - 1$ なら,	$x_2 = N_x - 1$
$y_1 < 0$ なら,	$y_1 = 0$
$y_2 > N_y - 1$ なら,	$y_2 = N_y - 1$

プログラムを始める前に良く考えることが重要！！



危険な不正アクセス！

```
#define WIDTH 8 // 画像の幅(ピクセル数)
#define HEIGHT 6 // 画像の高さ(ピクセル数)
配列 image[WIDTH][HEIGHT]
```

i = WIDTH

	i=0	1	2	3	4	5	6	7
j=0	1	0	0	0	0	0	0	0
j=1	0	1	0	0	0	0	0	0
j=2	0	0	1	0	0	0	0	0
j=3	0	0	0	1	0	0	0	0
j=4	0	0	0	0	1	0	0	0
j=5	0	0	0	0	0	1	0	0

j = HEIGHT →

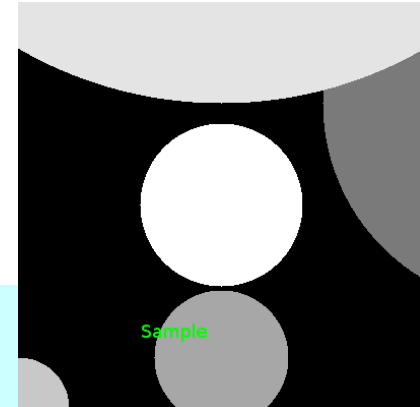
存在しない配列要素！
確保していないメモリ！

不正ア
クセス

運が良い時 ⇒ 何も問題は起きない
(ただし正解と一致したのは偶然)

問題が起きるときの症状

- ✓ 途中でプログラムが停止しているが、一見正常終了
- ✓ 不正アクセス等のエラーメッセージで終了
- ✓ 突然、コンソールウィンドウ(黒ウィンドウ)が閉じる



```
#include <stdio.h>
#include <stdlib.h>
#include "cglec.h"
```

```
void PaintCircle(Image img, int x0, int y0, int r,
{
// この関数を作成する. x0,y0 : 中心座標, r :
```

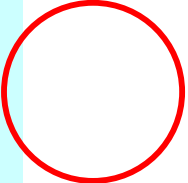
```
int main(void) // main()関数はこのまま使う. 修正
{
int Nx, Ny;
printf("画像の横方向ピクセル数は? "); scanf(
printf("画像の縦方向ピクセル数は? "); scanf(
unsigned char* data = (unsigned char*) malloc(s
if (data == NULL) //
{ printf("メモリエラー!!"); exit(0); } //
Image img = { (unsigned char*) data, Nx, Ny };
CglSetAll(img, 0);
PaintCircle(img, Nx/2, Ny/2, Nx/5, 255);
PaintCircle(img, 0, 0, Nx/8, 150);
PaintCircle(img, Nx/2, Ny/8, Nx/6, 100);
PaintCircle(img, 5*Nx/4, 3*Ny/4, Nx/2, 50);
PaintCircle(img, Nx/2, 7*Ny/4, Nx, 200);
CglSaveGrayBMP(img, "Circles.bmp");
free(data); //メモリ解放
}
```

どこでプログラムが落ちているか探す！

クリッピングとループ境界条件の注意

$x_1 < 0$ なら,	$x_1 = 0$
$x_2 > N_x - 1$ なら,	$x_2 = N_x - 1$
$y_1 < 0$ なら,	$y_1 = 0$
$y_2 > N_y - 1$ なら,	$y_2 = N_y - 1$

```
for (y = y1; y <= y2; y++)  
{  
    for (x = x1; x <= x2; x++)  
    {  
        if( (x-x0)*(x-x0) + (y-y0)*(y-y0) <= r*r)  
            *(img.Data + x*img.Ny + y) = g;  
    }  
}
```



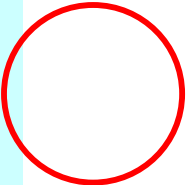
```
for (y = y1; y < y2; y++)  
{  
    for (x = x1; x < x2; x++)  
    {  
        if( (x-x0)*(x-x0) + (y-y0)*(y-y0) <= r*r)  
            *(img.Data + x*img.Ny + y) = g;  
    }  
}
```

理由: 不正アクセスは起きないが
端の1列が描画されない



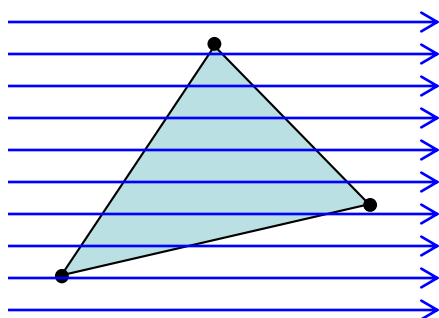
$x_1 < 0$ なら,	$x_1 = 0$
$x_2 > N_x$ なら,	$x_2 = N_x$
$y_1 < 0$ なら,	$y_1 = 0$
$y_2 > N_y$ なら,	$y_2 = N_y$

```
for (y = y1; y < y2; y++)  
{  
    for (x = x1; x < x2; x++)  
    {  
        if( (x-x0)*(x-x0) + (y-y0)*(y-y0) <= r*r)  
            *(img.Data + x*img.Ny + y) = g;  
    }  
}
```



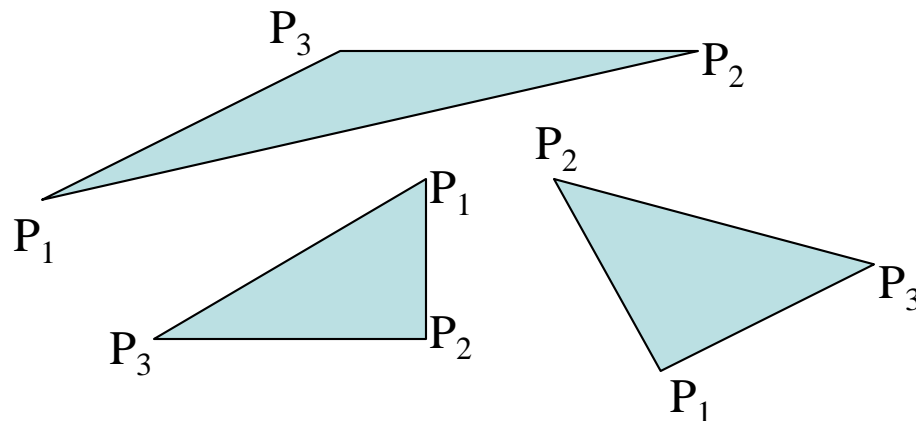
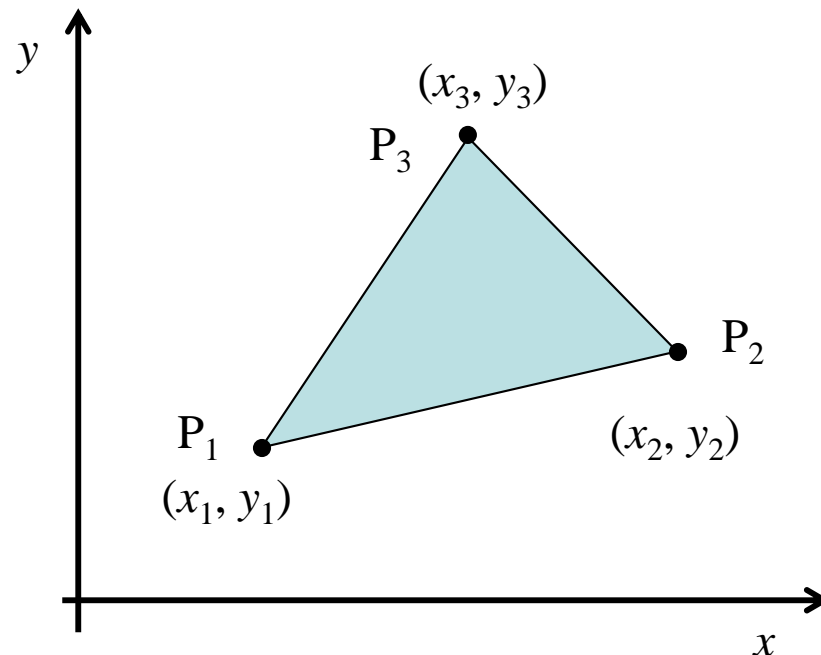
三角形の塗りつぶし(1)

3点 P_1, P_2, P_3 の座標を与えたとき、この3点を頂点とする三角形を塗りつぶす関数を考える



ラスタスキャンを行う

- ✓ 内側の判定をどうするか？
- ✓ クリッピングをどうするか？



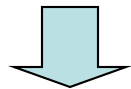
三角形の塗りつぶし(3) — 正領域・負領域

2点を通る直線の方程式(陽関数)

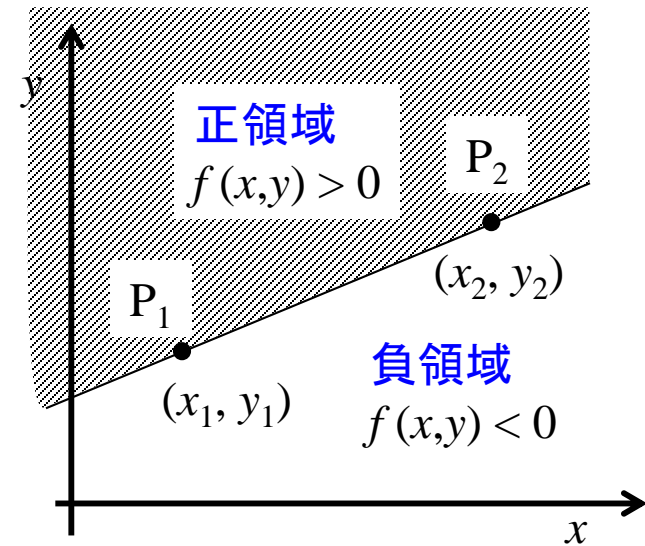
$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$

2点を通る直線の方程式(陰関数)

$$f(x, y) = (y - y_1) - \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) = 0$$

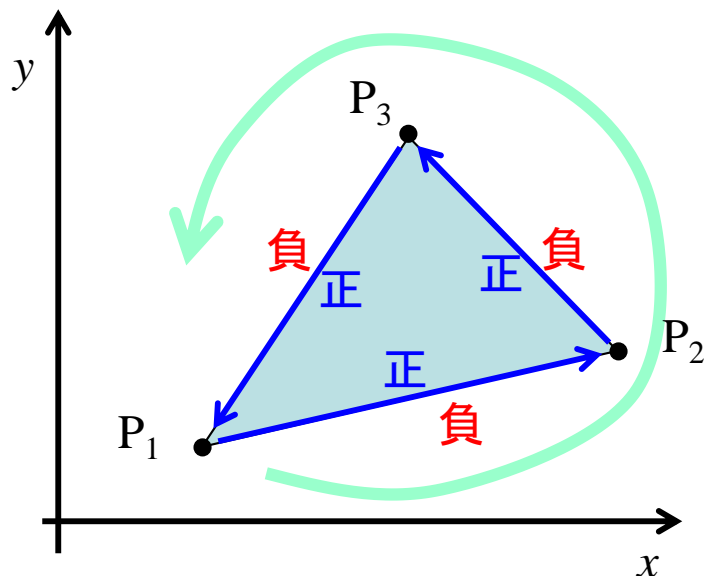


$x_2 = x_1$ で発散



$$f(x, y) = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

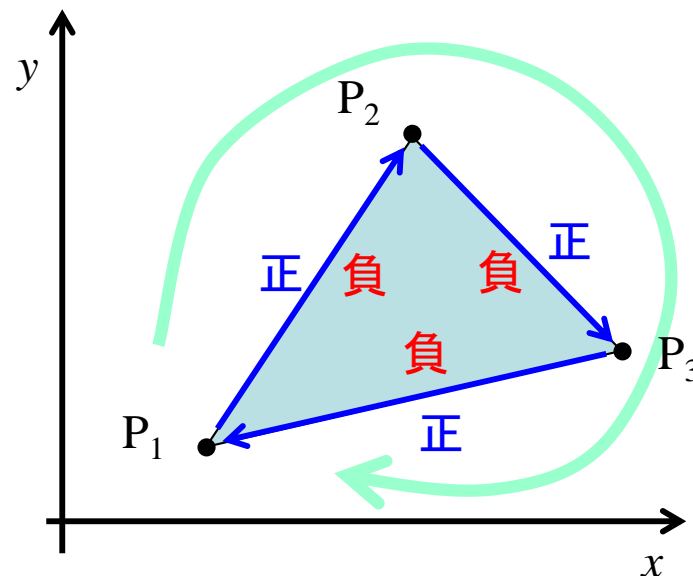
三角形の塗りつぶし(7) — 順序による差異



- ✓ 直線 P_1P_2 について正領域
- ✓ 直線 P_2P_3 について正領域
- ✓ 直線 P_3P_1 について正領域

パターン I

反時計回り

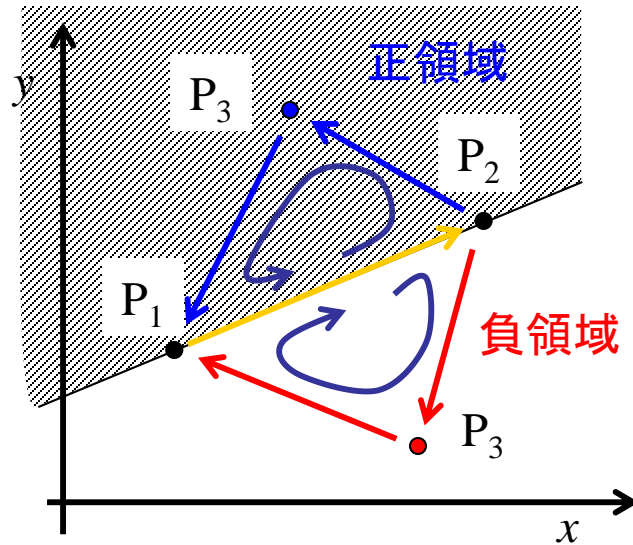


- ✓ 直線 P_1P_2 について負領域
- ✓ 直線 P_2P_3 について負領域
- ✓ 直線 P_3P_1 について負領域

パターン II

時計回り

三角形の塗りつぶし(8) 一回る方向の判別



直線 P_1P_2 に対して

もし点 P_3 が**正領域**にあれば, パターン I

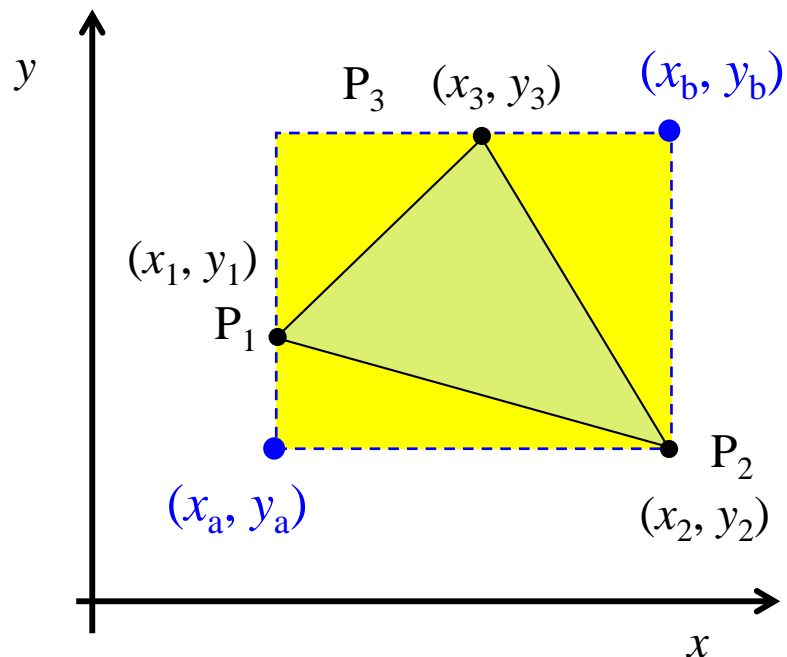
→ 3辺に対して全て**正領域**の条件で塗りつぶし

もし点 P_3 が**負領域**にあれば, パターン II

→ 3辺に対して全て**負領域**の条件で塗りつぶし

三角形の塗りつぶし(9) – クリッピング

復習

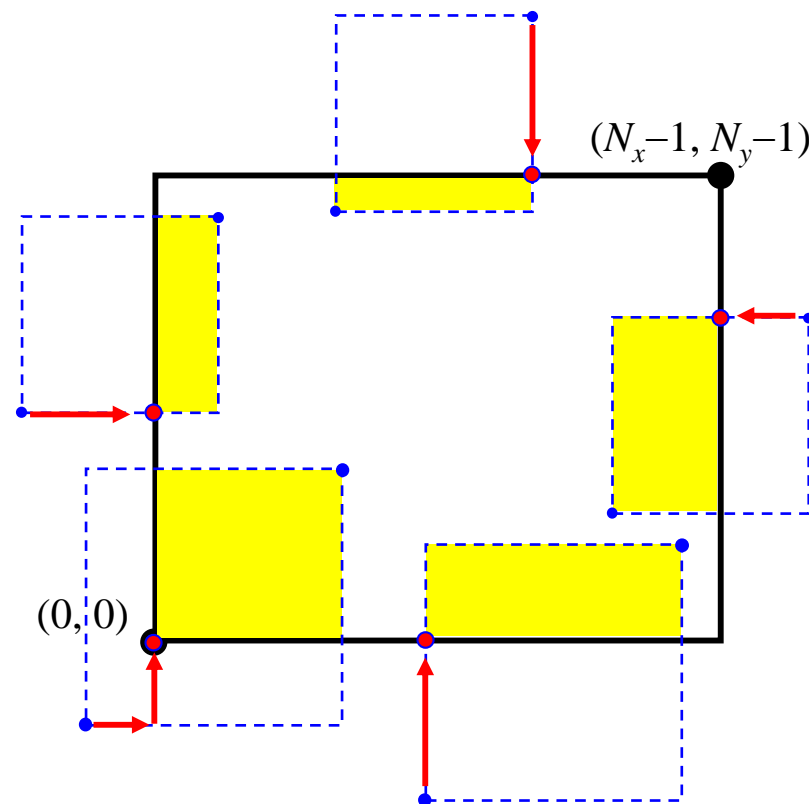


$x_a = [x_1, x_2, x_3 \text{ の中で最も小さな値}]$

$y_a = [y_1, y_2, y_3 \text{ の中で最も小さな値}]$

$x_b = [x_1, x_2, x_3 \text{ の中で最も大きな値}]$

$y_b = [y_1, y_2, y_3 \text{ の中で最も大きな値}]$



三角形の塗りつぶし — まとめ

- Step 1 三角形を囲む領域(x_a, y_a), (x_b, y_b)を求める
- Step 2 三角形を囲む領域が画像の外にはみ出している場合のクリッピング処理を行う.
- Step 3 3点の順番がパターン I (反時計回り)かパターン II (時計周り)かを判定する
- Step 4 ラスタスキャンをおこなう. ただし,
 - (i) 3点の順番がパターン I の場合は, 3辺について正領域を塗りつぶす
 - (ii) 3点の順番がパターン II の場合は, 3辺について負領域を塗りつぶす

基本課題5

復習

3点 (x_1, y_1) , (x_2, y_2) , (x_3, y_3) を頂点とする三角形をグレーレベル g で塗りつぶす関数 `PaintTriangle()` を作成せよ. 作成した関数と下記の`main()`を組み合わせることで実行例と同じ画像を得よ. なお関数`LineFunc()`を有効に活用すること.

Report5-1

$$f(x, y) = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

```
#include <stdlib.h>
#include <stdio.h>
#include "cglec.h"
```

```
int LineFunc(int x, int y, int x1, int y1, int x2, int y2) //点(x,y)が正領域なら正值,
{ return (x2 - x1)*(y - y1) - (y2 - y1)*(x - x1); } //負領域なら負値を返す関数
```

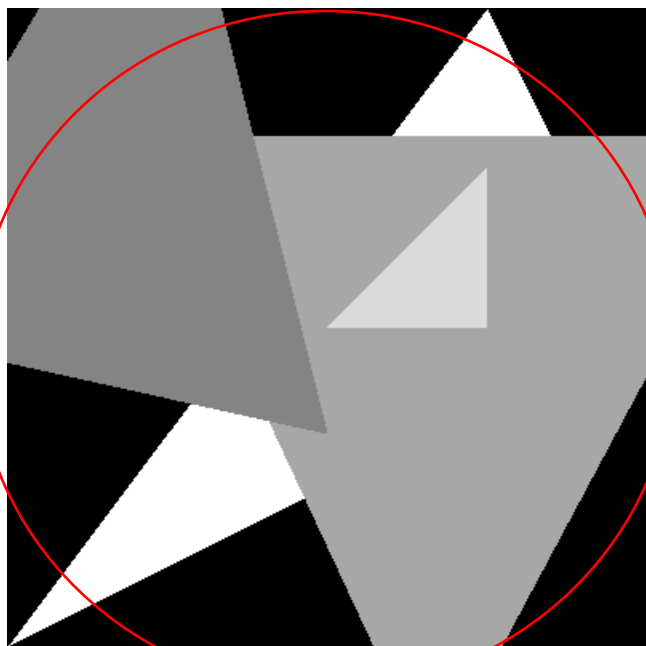
```
void PaintTriangle(Image img, int x1, int y1, int x2, int y2, int x3, int y3, int g)
{ /* この関数を作成せよ */ }
```

```
int main(void)
{
    int Nx, Ny;
    printf("画像の横方向ピクセル数は? "); scanf("%d", &Nx);
    printf("画像の縦方向ピクセル数は? "); scanf("%d", &Ny);
    unsigned char* data = (unsigned char*)malloc(sizeof(unsigned char) * Nx * Ny);
    if (data == NULL)
    { printf("メモリエラー!!"); exit(0); }
    Image img = { (unsigned char*)data, Nx, Ny };
    CglSetAll(img, 0);
    PaintTriangle(img, 0, 0, Nx - 1, Ny / 2, 3 * Nx / 4, Ny - 1, 255);
    PaintTriangle(img, Nx / 5, 4 * Ny / 5, Nx / 5 + Nx, 4 * Ny / 5, 2 * Nx / 3, -Ny / 5, 100);
    PaintTriangle(img, Nx / 2, Ny / 2, 3 * Nx / 4, 3 * Ny / 4, 3 * Nx / 4, Ny / 2, 180);
    PaintTriangle(img, Nx / 2, Ny / 3, Nx / 4, 4 * Ny / 3, -Nx / 4, Ny / 2, 60);
    CglSaveGrayBMP(img, "Triangles.bmp");
}
```

基本課題5 実行例

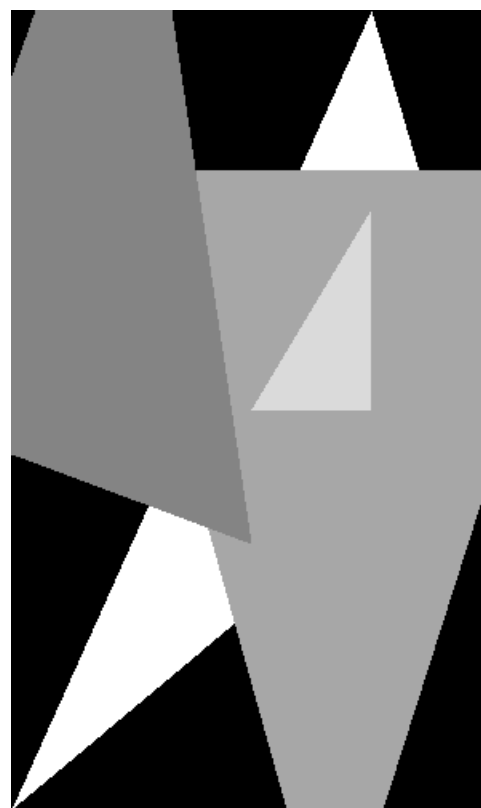
(1) 実行結果 (画面コピー)

画像の横方向ピクセル数は？ 400
 画像の縦方向ピクセル数は？ 400
 続行するには何かキーを押してください . . .



(2) 実行結果(BMPファイル)

画像の横方向ピクセル数は？ 300
 画像の縦方向ピクセル数は？ 500
 続行するには何かキーを押してください . . .



- ・ (1)と(2)の両方を提出！
- ・ 実行例を二つ以上提出！

基本課題5 解答例(1)

A君解答

```
void PaintTriangle(Image img, int x1, int y1, int x2, int y2, int x3, int y3, int g)
{
    int x, y, xa=x1, ya=y1, xb=x1, yb=y1;
    if(xa>x2) xa = x2; // xa = [x1, x2, x3のなかで最小値]
    if(xa>x3) xa = x3;
    if(ya>y2) ya = y2; // ya = [y1, y2, y3の中の最小値]
    if(ya>y3) ya = y3;
    if(xb<x2) xb = x2; // xb = [x1, x2, x3の中の最大値]
    if(xb<x3) xb = x3;
    if(yb<y2) yb = y2; // yb = [y1, y2, y3の中no最大値]
    if(yb<y3) yb = y3;

    if(xa<0) xa=0;
    if(ya<0) ya=0;
    if(xb>img.Nx) xb=img.Nx;
    if(yb>img.Ny) yb=img.Ny;

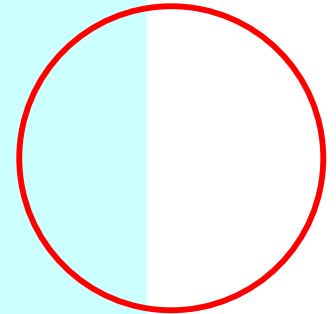
    if(g<0) g=0;
    if(g>255) g=255;
```

次ページへ続く

基本課題5 解答例(2)

```
if( LineFunc(x3, y3, x1, y1, x2, y2) > 0 )//時計回り
{
    //(i) 3点の順番がパターン I の場合は, 3辺について正領域を塗りつぶす
    for(y=ya; y<yb; y++)
    {
        for(x=xa; x<xb; x++)
        {
            if( LineFunc(x, y, x1, y1, x2, y2)>0
                && LineFunc(x, y, x2, y2, x3, y3) >0
                && LineFunc(x, y, x3, y3, x1, y1) >0 )
                *(img.Data+x*img.Ny+y) = g;
        }
    }
}

else //反時計回り
{
    //(ii) 3点の順番がパターン II の場合は, 3辺について負領域を塗りつぶす
    for(y=ya; y<yb; y++)
    {
        for(x=xa; x<xb; x++)
        {
            if( LineFunc(x, y, x1, y1, x2, y2)<0
                && LineFunc(x, y, x2, y2, x3, y3) <0
                && LineFunc(x, y, x3, y3, x1, y1) <0 )
                *(img.Data+x*img.Ny+y) = g;
        }
    }
}
```



基本課題5 解答例(3)

B君解答

```
void PaintTriangle(Image img, int x1, int y1, int x2, int y2, int x3, int y3, int g) {  
  
    /* この関数を作成せよ */  
  
    int xa = x1;    if (xa > x2) xa = x2;        if (xa > x3) xa = x3;        //xの最小値  
    int ya = y1;    if (ya > y2) ya = y2;        if (ya > y3) ya = y3;        //yの最小値  
    int xb = x1;    if (xb < x2) xb = x2;        if (xb < x3) xb = x3;        //xの最大値  
    int yb = y1;    if (yb < y2) yb = y2;        if (yb < y3) yb = y3;        //yの最大値  
  
    if (xa < 0) xa = 0;        if (ya < 0) ya = 0;        //はみ出しの判定  
    if (xb > img.Nx) xb = img.Nx;    if (yb > img.Ny) yb = img.Ny;  
  
    int rot;    //rot = 1 正領域 (左回転)  rot = -1 領域 (右回転)  
    if (LineFunc(x3, y3, x1, y1, x2, y2) > 0) rot = 1; else rot = -1;  
  
    int x, y;  
    for (x = xa; x < xb; x++) {  
        for (y = ya; y < yb; y++) {  
            if (rot * LineFunc(x, y, x1, y1, x2, y2) >= 0  
                && rot * LineFunc(x, y, x2, y2, x3, y3) >= 0  
                && rot * LineFunc(x, y, x3, y3, x1, y1) >= 0)  
                *(img.Data + x*img.Ny + y) = g;  
        }  
    }  
}
```

良くできました!

基本課題5 解答例(4)

C君解答

```
void PaintTriangle(Image img, int x1, int y1, int x2, int y2, int x3, int y3, int g)
{
    if (g < 0)      g = 0;
    if (g > 255)    g = 255;

    //---3点を反時計回りに領域判定するように補正する---//
    if (LineFunc(x3, y3, x1, y1, x2, y2) < 0) {
        //---時計回りなら(x2, y2)と(x3, y3)を入れ替え---//
        int tmpX = x2, tmpY = y2;
        x2 = x3; y2 = y3;
        x3 = tmpX; y3 = tmpY;
    }

    int xRangeMin = ((minOf(x1, x2, x3) > 0) ? minOf(x1, x2, x3) : 0);
    int yRangeMin = ((minOf(y1, y2, y3) > 0) ? minOf(y1, y2, y3) : 0);
    int xRangeMax = ((maxOf(x1, x2, x3) + 1 < img.Nx) ? maxOf(x1, x2, x3) + 1 : img.Nx);
    int yRangeMax = ((maxOf(y1, y2, y3) + 1 < img.Ny) ? maxOf(y1, y2, y3) + 1 : img.Ny);

    //---三角形の描画---//
    int x, y;
    for (y = yRangeMin; y < yRangeMax; y++) {
        for (x = xRangeMin; x < xRangeMax; x++) {
            if (isInTriangle(x, y, x1, y1, x2, y2, x3, y3))
                *(img.Data + img.Ny * x + y) = g;
        }
    }
}
```

条件演算子

(if文を簡潔に書く手法)

<式1> ? <式2> : <式3>

この式は、<式1>が真なら<式2>の値になり、偽なら<式3>の値になる。

minOf()とmaxOf()は、それぞれ引数の最大、最小を求める関数として、C君がソース中に定義している

良くできました!

基本課題5 解答例(5)

D君解答

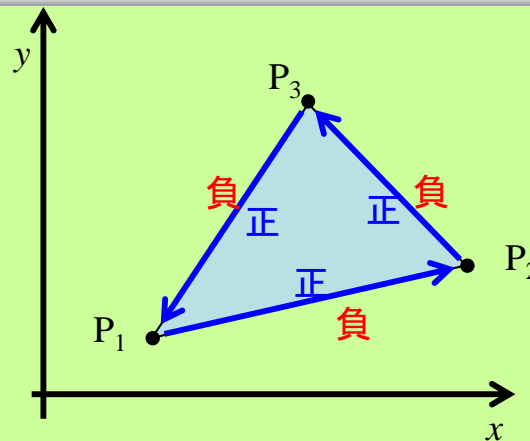
```
for (X=Xmin; X<Xmax; X++)  
{  
    for (Y=Ymin; Y<Ymax; Y++)  
    {  
        L1=LineFunc (X, Y, x1, y1, x2, y2);  
        L2=LineFunc (X, Y, x2, y2, x3, y3);  
        L3=LineFunc (X, Y, x3, y3, x1, y1);  
  
        if ((L1>0 && L2>0 && L3>0) || (L1<0 && L2<0 && L3<0))  
            *(img.Data+X*img.Ny+Y)=g;  
    }  
}
```

点の順番(パターン)の判定
を行わない!

良くでき
ました!

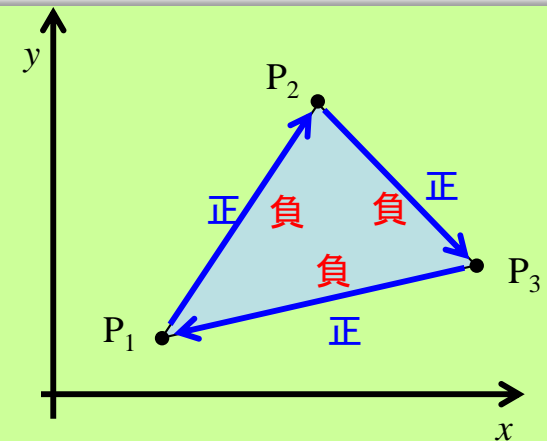
全て正領域
または
全て不領域
であれば, 3点の順
番に関係なく内側と
判定できる.

L1, L2, L3の一つで
も異なった符号であ
れば, それは三角形
の外部



- ✓ 直線 P_1P_2 について正領域
- ✓ 直線 P_2P_3 について正領域
- ✓ 直線 P_3P_1 について正領域

パターン I



- ✓ 直線 P_1P_2 について負領域
- ✓ 直線 P_2P_3 について負領域
- ✓ 直線 P_3P_1 について負領域

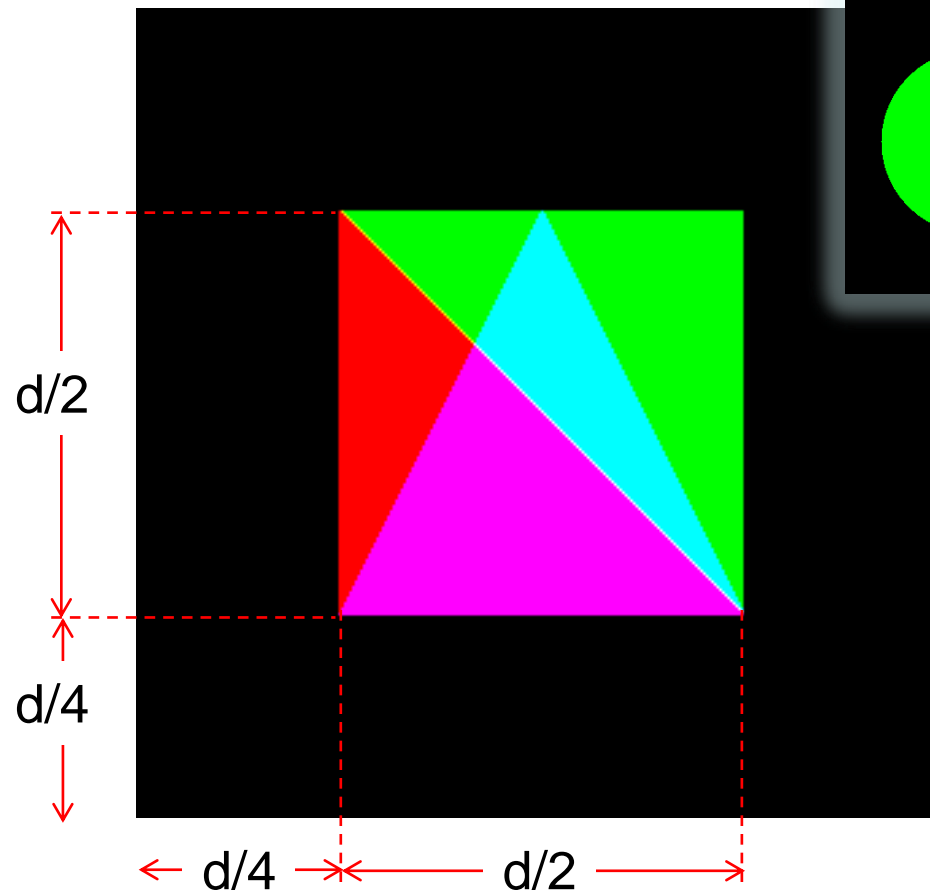
パターン II

発展課題5

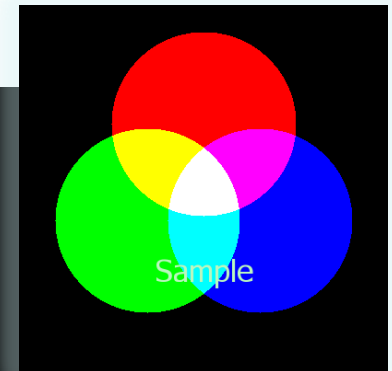
復習

PaintTriangle()関数を用いて、次のような画像を作り出すプログラムを作成せよ.

画像の横方向ピクセル数は? 300
画像の縦方向ピクセル数は? 300
続行するには何かキーを押してください . . .



d: 短辺



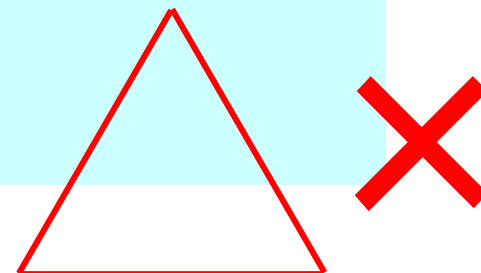
発展課題5 解答例

C君解答

```
int main(void)
{
    int Nx, Ny;
    int d=0;
    printf("画像の横方向ピクセル数は?"); scanf("%d", &Nx);
    printf("画像の縦方向ピクセル数は?"); scanf("%d", &Ny);
    unsigned char* red = (unsigned char*) malloc(sizeof(unsigned char) * Nx * Ny);
    unsigned char* green = (unsigned char*) malloc(sizeof(unsigned char) * Nx * Ny);
    unsigned char* blue = (unsigned char*) malloc(sizeof(unsigned char) * Nx * Ny);
    if(Nx>Ny)    d=Ny;
    else        d=Nx;

    Image img_red = { (unsigned char*) red, Nx, Ny };
    Image img_green = { (unsigned char*) green, Nx, Ny };
    Image img_blue = { (unsigned char*) blue, Nx, Ny };

    CglSetAll(img_red, 0);
    CglSetAll(img_green, 0);
    CglSetAll(img_blue, 0);
    PaintTriangle(img_red, d/4, d/4, 3*d/4, d/4, d/4, 3*d/4, 255);
    PaintTriangle(img_green, d/4, 3*d/4, 3*d/4, d/4, 3*d/4, 3*d/4, 255);
    PaintTriangle(img_blue, d/4, d/4, 3*d/4, d/4, d/2, 3*d/4, 255);
    CglSaveColorBMP(img_red, img_green, img_blue, "Triangles.bmp");
}
```

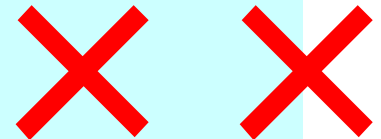


発展課題5 解答例

D君解答

```
int main(void)
{
    int Nx, Ny, Nt;
    printf("画像の横方向ピクセル数は?"); scanf("%d", &Nx);
    printf("画像の縦方向ピクセル数は?"); scanf("%d", &Ny);
    unsigned char* data = (unsigned char*) malloc(sizeof(unsigned char) * Nx * Ny);
    if (data == NULL)
    { printf("メモリエラー!!"); exit(0); }
    Image img = { (unsigned char*) data, Nx, Ny };

    unsigned char red[400][400];
    unsigned char green[400][400];
    unsigned char blue[400][400];
    Image img_red = {(unsigned char*)red, Nx, Ny};
    Image img_green = {(unsigned char*)green, Nx, Ny};
    Image img_blue = {(unsigned char*)blue, Nx, Ny};
    CglSetAll(img, 0);
    CglSetAll(img_red, 0);
    CglSetAll(img_green, 0);
    CglSetAll(img_blue, 0);
    Nt=Nx;
    if(Nx>Ny)
        Nt=Ny;
    PaintTriangle(img_red, Nt/4, Nt/4, 3*Nt/4, Nt/4, Nt/4, 3*Nt/4, 255);
    PaintTriangle(img_green, Nt/4, 3*Nt/4, 3*Nt/4, Nt/4, 3*Nt/4, 3*Nt/4, 255);
    PaintTriangle(img_blue, Nt/4, Nt/4, 3*Nt/4, Nt/4, Nt/2, 3*Nt/4, 255);
    CglSaveColorBMP(img_red, img_green, img_blue, "ColorTriangles.bmp");
}
```

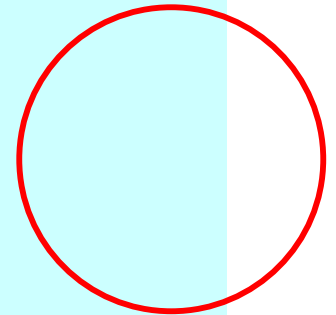


発展課題5 解答例

E君解答

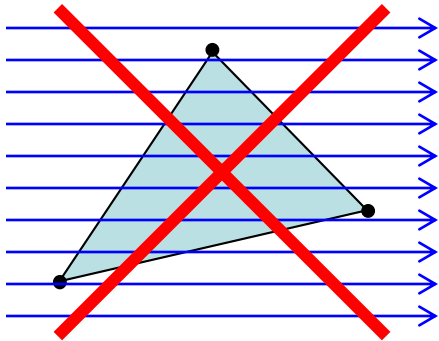
```
int main(void)
{
    int Nx, Ny, d, s;
    printf("画像の横方向ピクセル数は?"); scanf("%d", &Nx);
    printf("画像の縦方向ピクセル数は?"); scanf("%d", &Ny);
    if(Nx>Ny)
    { d=Ny; }
    else
    { d=Nx; }
    s=d*3;
    unsigned char* blu = (unsigned char*) malloc(sizeof(unsigned char) * Nx * Ny);
    unsigned char* red = (unsigned char*) malloc(sizeof(unsigned char) * Nx * Ny);
    unsigned char* gre = (unsigned char*) malloc(sizeof(unsigned char) * Nx * Ny);

    if (red == NULL || gre == NULL || blu == NULL)
    { printf("メモリエラー!!"); exit(0); }
    Image img_aoi = { (unsigned char*) blu, Nx, Ny };
    Image img_aka = { (unsigned char*) red, Nx, Ny };
    Image img_mid = { (unsigned char*) gre, Nx, Ny };
    CglSetAll(img_aoi, 0);
    CglSetAll(img_aka, 0);
    CglSetAll(img_mid, 0);
    PaintTriangle(img_mid, d/4, d/4, s/4, d/4, d/2, s/4, 255); //青色の三角
    PaintTriangle(img_aoi, d/4, s/4, d/4, d/4, s/4, d/4, 255); //赤色の三角
    PaintTriangle(img_aka, d/4, s/4, s/4, s/4, s/4, d/4, 255); //緑色の三角
    CglSaveColorBMP(img_aoi, img_aka, img_mid, "Triangles.bmp");
    free(blu);
    free(red);
    free(gre);
}
```

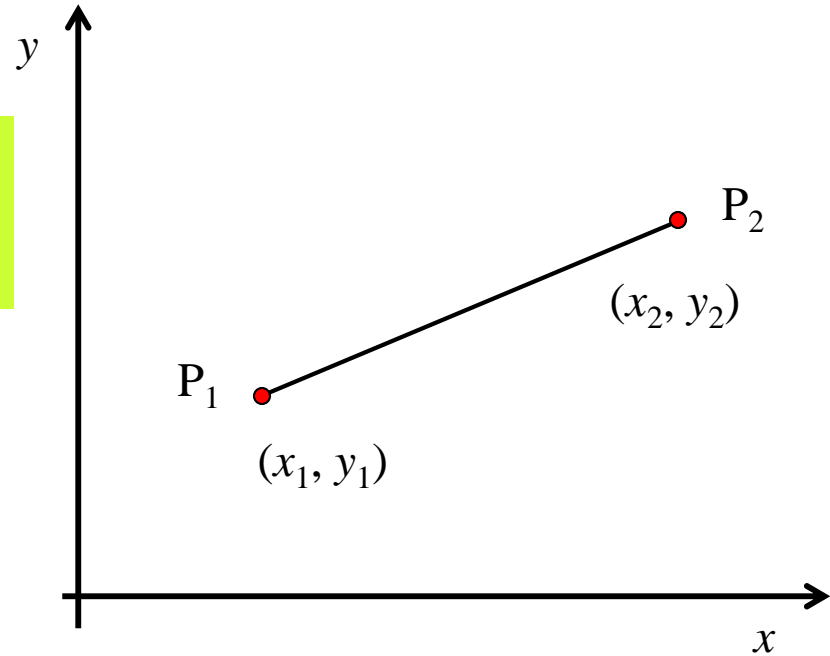


直線の描画

2点 P_1 , P_2 の座標を与えたとき、この2点を結ぶ線分を描画する関数を考える



ラスタスキャン



$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$

直線の描画(1): 単純なアルゴリズム

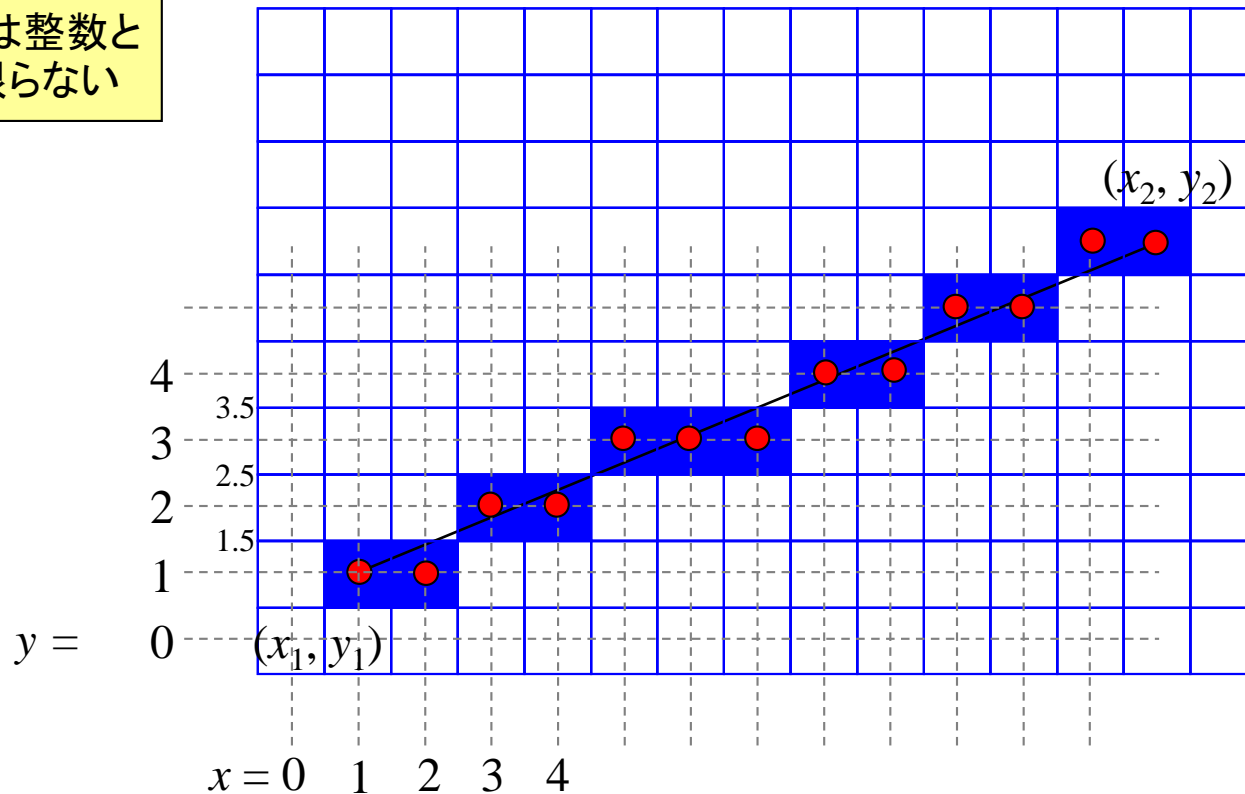
$$y = m(x - x_1) + y_1$$

$$m \equiv \frac{y_2 - y_1}{x_2 - x_1}$$

傾きは整数とは限らない

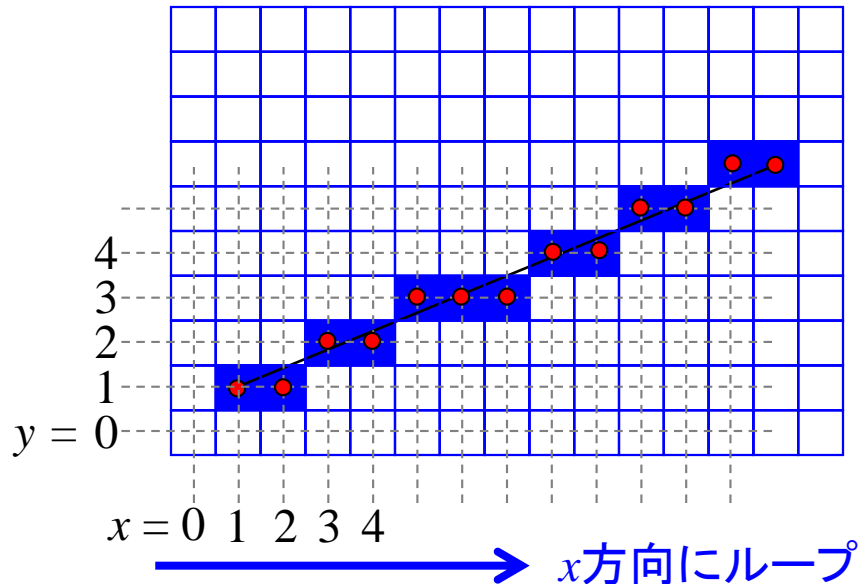
- (1) x を1増やす
- (2) $y' = m(x - x_1) + y_1$ を求める
- (3) y' を四捨五入して y を求める
- (4) (x, y) のピクセルのグレイレベルを変える

y' は整数とは限らない



→ x 方向にループ

直線の描画(2): 単純なアルゴリズム



$$y = m(x - x_1) + y_1$$

$$m \equiv \frac{y_2 - y_1}{x_2 - x_1}$$

傾きは整数とは限らない

y' は整数とは限らない

- (1) x を1増やす
- (2) $y' = m(x - x_1) + y_1$ を求める
- (3) y' を四捨五入して y を求める
- (4) (x, y) のピクセルのグレイレベルを変える

Example6-1

```
void DrawLine0(Image img, int x1, int y1, int x2, int y2, int g)
{
    // とりあえずクリッピングやパラメータチェックはやらない
    double m = (double) (y2 - y1) / (x2 - x1); // 直線の傾き
    int x, y;
    for (x = x1; x <= x2; x++)
    {
        y = (int) (m * (x - x1) + y1 + 0.5); // 四捨五入して整数に変換
        *(img.Data + x*img.Ny + y) = g;
    }
}
```

(double) は double 型へのキャスト。割り算で少数以下が切り捨てられないようにする。

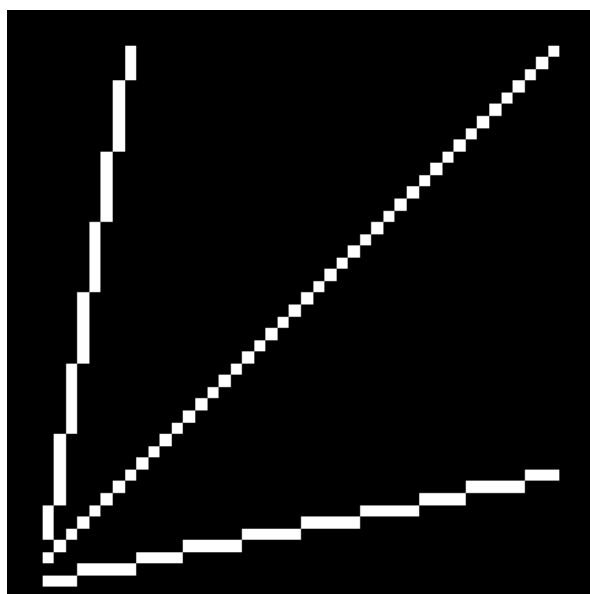
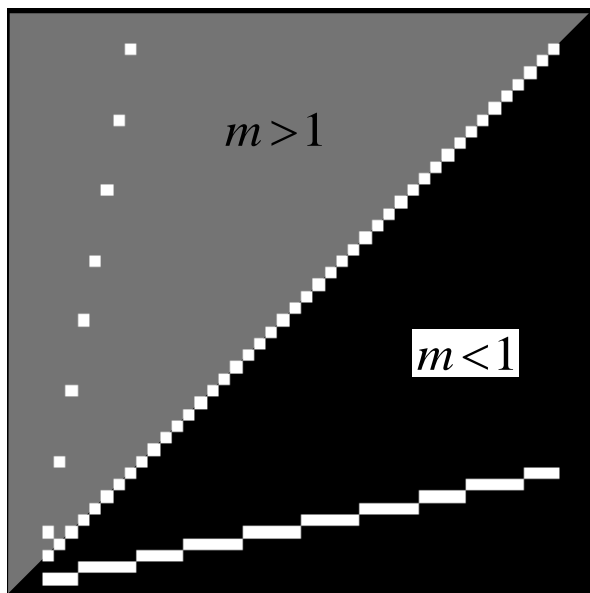
$x1 == x2$?
ゼロ除算

(int) は int 型へのキャスト。少数以下は切り捨てられる

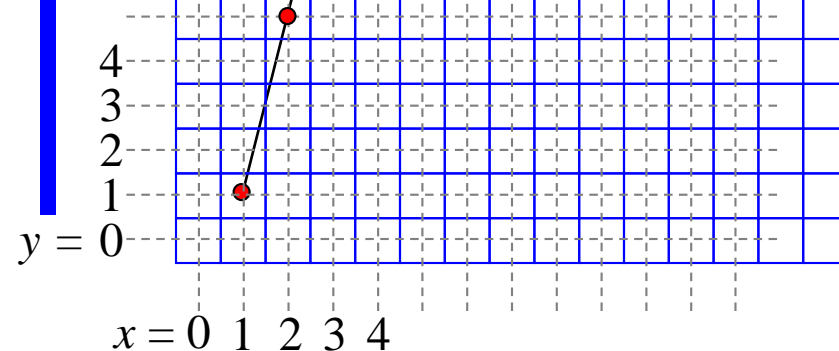
0.5 を足して少数以下を切り捨てると、四捨五入

基礎プログラミング
講習第13回

直線の描画: 単純なアルゴリズムの結果

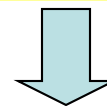


y方向に
ループ



対処法

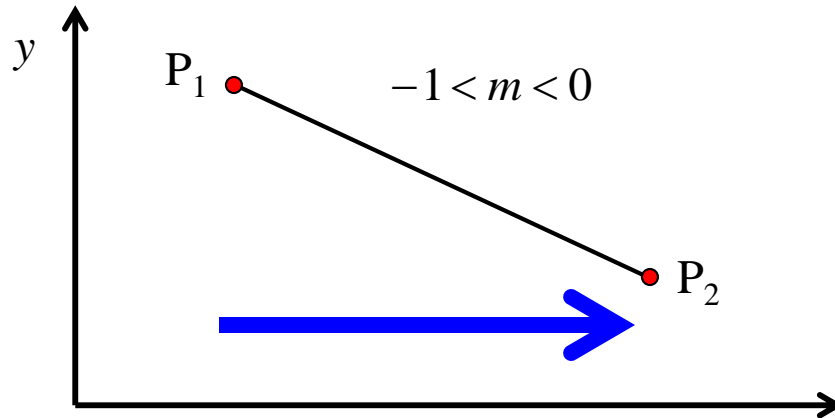
$m > 1$ と $m < 1$ では別に扱う



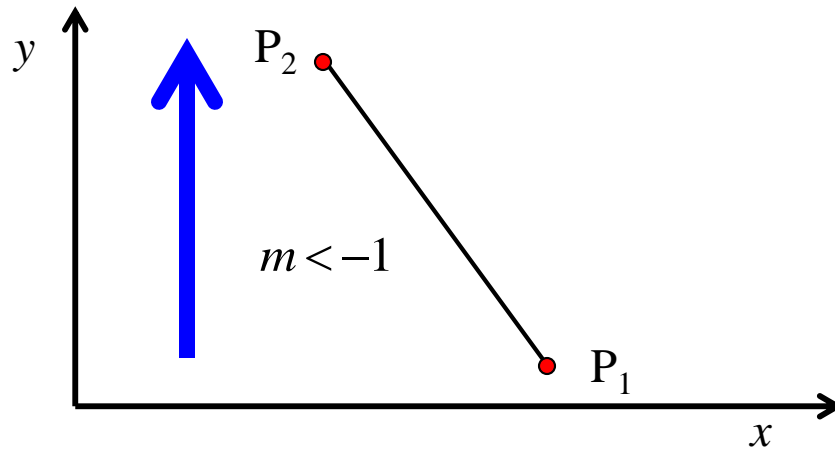
$m > 1$ の場合は x と y の役割を逆転させる

$$x' = (y - y_1) / m + x_1$$

直線の描画(パラメータチェック)

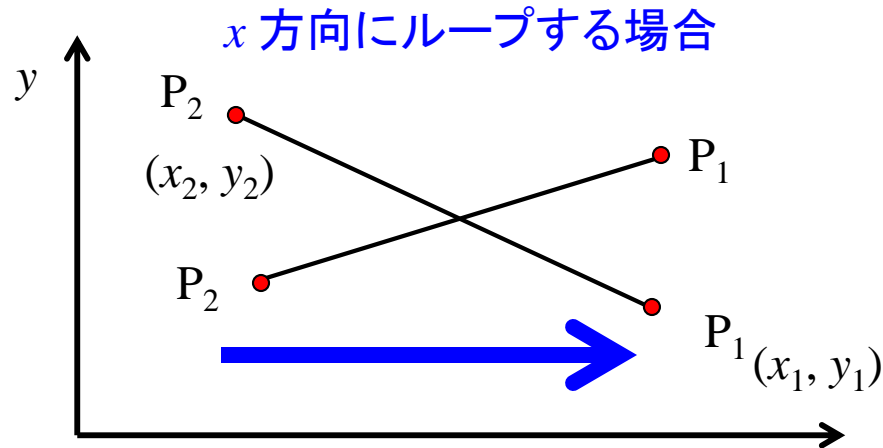


- (i) $|m| < 1$ の場合
 $\Rightarrow x$ 方向にループ



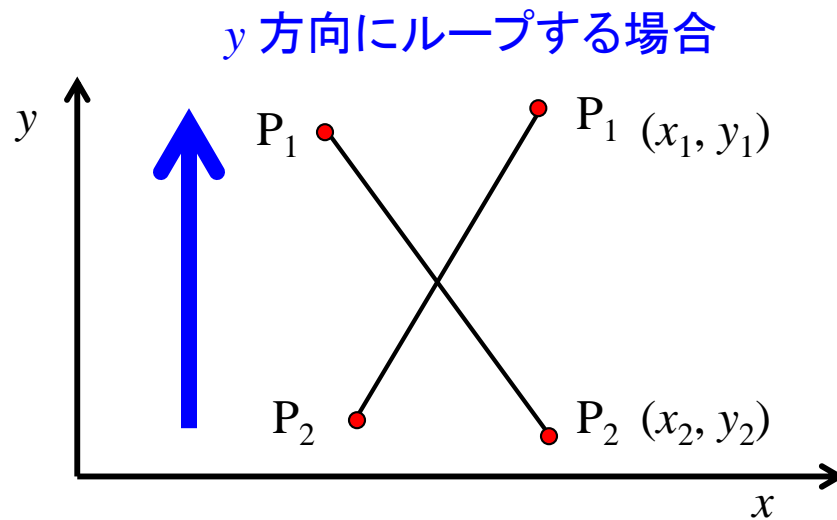
- (ii) $|m| > 1$ の場合
 $\Rightarrow y$ 方向にループ

直線の描画(パラメータチェック)



$x_1 > x_2$ の場合

$\Rightarrow P_1$ と P_2 を入れ替え



$y_1 > y_2$ の場合

$\Rightarrow P_1$ と P_2 を入れ替え

直線描画アルゴリズムのまとめ

傾き m をチェック

m の計算時にゼロ除算
が発生しないように！

$$m \equiv \frac{y_2 - y_1}{x_2 - x_1}$$

◆ $|m| \leq 1$ であれば

- x_1 と x_2 を比較
- $x_1 > x_2$ なら P_1 と P_2 の座標値を入れ替え
- x_1 から x_2 までループする

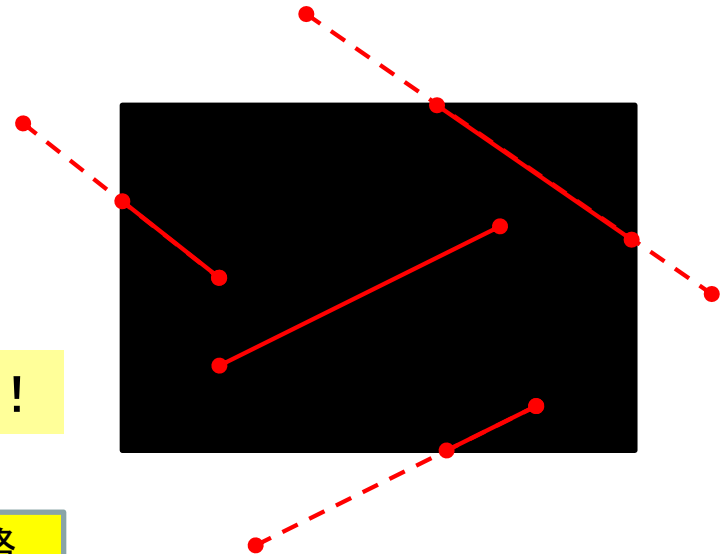
もしも $x_1 = x_2$ だったら, 垂
直方向の直線

◆ $|m| > 1$ であれば,

- y_1 と y_2 を比較
- $y_1 > y_2$ なら P_1 と P_2 の座標値を入れ替え
- y_1 から y_2 までループする

注) 直線のクリッピングは複雑！

この授業では省略



基本課題6

点 $P_1(x_1, y_1)$ と点 $P_2(x_2, y_2)$ の間にグレーレベル g の線分を描画する`DrawLine()`関数を作成せよ。作成した関数と下記の`main()`を組み合わせる実行結果と同じ画像を得よ。なお、この関数ではクリッピングは考慮しなくて良い。
(点 P_1 と点 P_2 は常に画像の範囲内にあるものとする) ただし、ゼロ除算を発生させないこと。

Report6-1

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "cglec.h"
```

```
void DrawLine(Image img, int x1, int y1, int x2, int y2, int g)
{ /* この部分をプログラム */ }
```

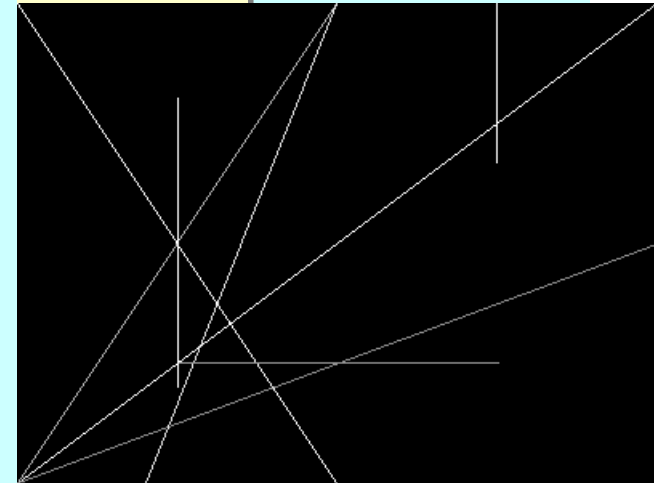
```
int main(void)
{   int Nx, Ny;
    printf("画像の横方向ピクセル数は? "); scanf("%d", &Nx);
    printf("画像の縦方向ピクセル数は? "); scanf("%d", &Ny);
    unsigned char* data = (unsigned char*)
                           malloc(sizeof(unsigned char) * Nx * Ny);

    if (data == NULL)
    {   printf("メモリエラー!!");   exit(0);   }
    Image img = { (unsigned char*) data, Nx, Ny };
    CglSetAll(img, 0); // imgをグレイレベル0でクリアする
    DrawLine(img, 0, 0, Nx - 1, Ny - 1, 255);
    DrawLine(img, 0, 0, Nx - 1, Ny / 2 - 1, 70);
    DrawLine(img, 0, 0, Nx / 2 - 1, Ny - 1, 128);
    DrawLine(img, 0, Ny - 1, Nx / 2 - 1, 0, 255);
    DrawLine(img, Nx / 2 - 1, Ny - 1, Nx / 5, 0, 200);
    DrawLine(img, 3 * Nx / 4 - 1, Ny - 1, 3 * Nx / 4 - 1, 2 * Ny / 3, 255);
    DrawLine(img, Nx / 4, Ny / 4, 3 * Nx / 4, Ny / 4, 80);
    DrawLine(img, Nx / 4, Ny / 5, Nx / 4, 4 * Ny / 5, 255);
    free(data);
}
```

・ コンソール画面と画像の両方を提出！

・ 実行例を二つ提出！

実行結果



画像の横方向ピクセル数は? 400
画像の縦方向ピクセル数は? 300
続行するには何かキーを押してください . . .

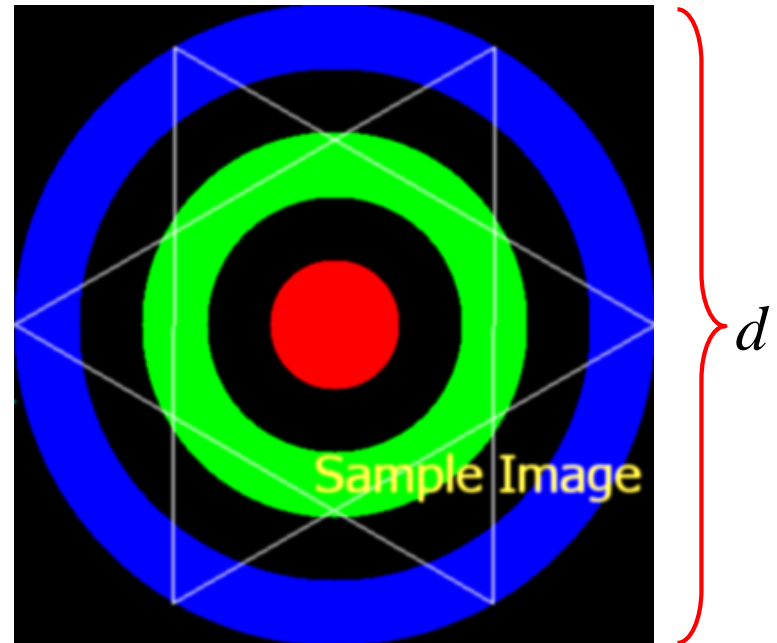
発展課題6

DrawLine()関数とPaintCircle()関数を用いて、次のような画像を作り出すプログラムを作成せよ。

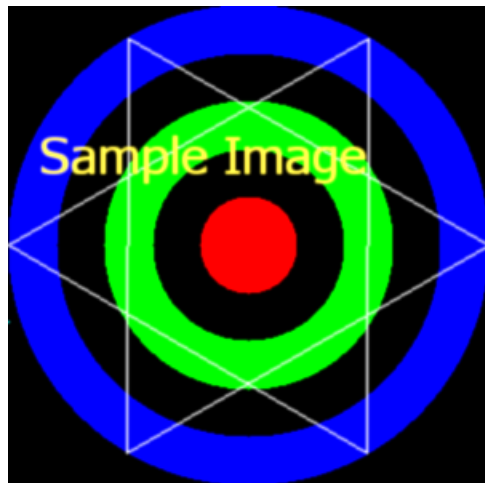
注) PaintCircle()関数の代わりにcglecに組み込まれているCglPaintCircle()を用いても良い

画像の縦と横方向ピクセル数は？ 400
続行するには何かキーを押してください . . .

- 画像の縦と横のピクセル数は同じ
- 画像の一边 d として,
 - 青円の外径は d , 内径は $4d/5$
 - 緑円の外径は $3d/5$, 内径は $2d/5$
 - 赤円の直径は $d/5$
- 星型六角形(六芒星)の半径は $d/2$ で, 線の色は白色.
- 「Sample Image」の文字は不要



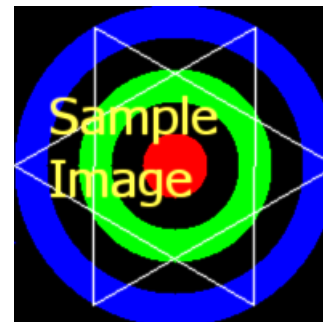
発展課題 実行例



画像の縦と横方向ピクセル数は？ 300
続行するには何かキーを押してください . . .

- ・ コンソール画面と画像の両方を提出！
- ・ 実行例を二つ提出！

画像の縦と横方向ピクセル数は？ 200
続行するには何かキーを押してください . . .



できるだけエレガントに短くプログラムしてみよう！