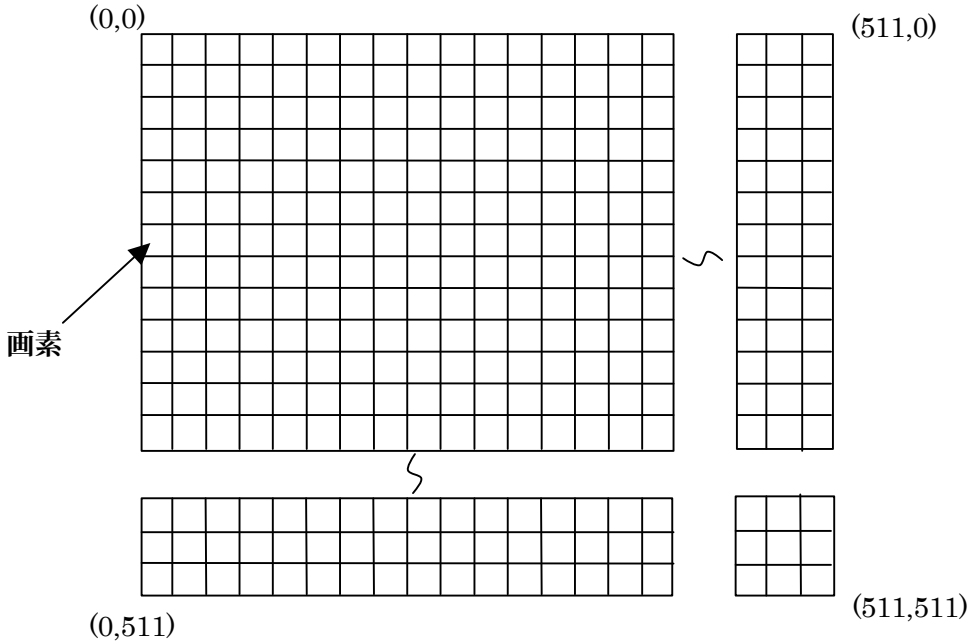
グラフィック

テキストではグラフィックは PGM フォーマットでファイルに出力したあとに、描画表示ソフトウェアによって表示をしています。

しかしながら MacOS X では X11 アプリケーションを直接用いる事によって、プログラムから簡単に図形を描画する事が可能となります。

グラフィックの画面は下図のように一般的に左上が原点となり、そのひとマスひとマスが画素と呼ばれます。グラフィック画面の表示サイズはモニターの解像度に依存してきますが、ここでは 512 画素×512 画素の画面を表示することとします(サイズの変更は可能)。



標準の C 言語にはグラフィックは含まれていません。仮にグラフィックが使えたとしても、それらの関数は処理系によって異なります。今回は X11R6 の関数を簡単に使えるようにしたライブラリーを使用しているので、その点は注意が必要です。

グラフィックの準備として HP より「graph.c」というファイルをダウンロードして、プログラムを作成するフォルダーに保存してください。そしてプログラムの始めに #include"graph.c" とグラフィック関数を使用するための宣言をします。

以下、使用可能な関数の説明となります。関数に渡す引数は全て int 型です。また戻り値はありません。今回用いるグラフィック関数はすべて G_????で始まります。

グラフィック関数一覧

G_start　　**機能　：グラフィックの初期化。最初に一度だけ呼ぶ必要があります。**
使用例：G_start();

G_end　　**機能　：グラフィックの終了。プログラムの最後に一度だけ呼びます。**
使用例：G_end();

G_cls()　　**機能　：グラフィック画面の消去。**
使用例：G_cls();

G_sleep()　　**機能　：グラフィック画面の停止。マウスの右ボタンで停止解除。**
使用例：G_sleep();

G_line　　**機能　：点**(ix,iy)**から点**(jx,jy)**に線分を引く。**
使用例：G_line(ix, iy, jx, jy);

G_pset　　**機能　：**(ix,iy)**に点を描画。**
使用例：G_pset(ix, iy);

G_circle　　**機能　：**(ix,iy)**を中心に半径** ir **の円を描く。**
使用例：G_circle(ix, iy, ir);
円の内側の塗りつぶしは色を指定した後 G_fcircle **を使います。**

G_box　　**機能　：左上点**(ix,iy)**から x 方向** jx**、y 方向** jy **の長さの四角形を描く。**(jx,jy **は正の整数)**
使用例：G_box(ix, iy, jx, jy);
四角の内側の塗りつぶしは色を指定した後 G_fbox **を使います。**

G_ellipse　　**機能　：左上点**(ix,iy)**から x 方向** jx**、y 方向** jy **の長さの四角形に内接する円を描く。**
使用例：G_ellipse(ix, iy, jx, jy);
楕円の内側の塗りつぶしは色を指定した後 G_fellipse **を使います。**

G_color　　**機能　：色の指定。**
使用例：G_color(“white”)； **，** G_color(“red”)； **・・・など**
色名として使える値は more /usr/lib/X11/rgb.txt **で参照してください**
(more **コマンドを終了させる場合、コントロールキー**(Ctrl)**と** C **キー**(c)**を同時に押す**)。

G_color_value　**機能　：色の指定。ｒ，ｇ，ｂの** 3 **原色で色を指定します。ただし各値は** 0≦red, green, blue≦255 **の範囲内です。**
使用例：G_color_value(255,255,255)； **，** G_color_value(255,0,0);

では実際にプログラムを入力して実行してみましょう。
以下のプログラムはテキストの sample29.c **を変更したものです。**

```
#include<stdio.h>
#include<math.h>

#include "graph.c"

#define N 512

main()
{
int k;
int x,y;

       G_start() ;
       for(x=-N/2 ; x< N/2 ; x++){
              for(y=-N/2 ; y< N/2 ; y++){
                     k=127.5*cos(3.14*(x*x+y*y)/ N)+127.5;
                     G_color_value(k,k,k);
                     G_pset(x+N/2,y+N/2);
              }
       }
       G_sleep();
       G_end();
}
```

graph.c **をインクルード**

必ずプログラムの最初に書く

プログラムの最後に書く

グラフィックを使う場合は「iTerm**」ではなく「**X11**」ターミナルを立ち上げます。**
コンパイルは以下のように行います

% gcc –L/usr/X11R6/lib filename.c –lm –lX11

実行は

% ./a.out

です。

MacOSX **において、画面キャプチャ(画面表示を、あたかも写真を取るように、画像データへと変換する方法)は以下のコマンド(命令)で可能となります。**

画面全体のキャプチャ
コマンド(Apple **キー)+シフトキー+**3
選択部分のみのキャプチャ
コマンド(Apple **キー)+シフトキー+**4

いずれの操作においても、png **フォーマットの画像データが作成されます。**

次のプログラムは、教科書 p57sample13_2.c モンテカルロ法による円周率計算プログラムを改変したものです。乱数によって発生した点が 1/4 円内の場合、関数 G_pset()を使って点を打ちます。

```
***プログラムここから***
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "graph.c"/*グラフィック利用*/
float myrand(){
float ans;
ans=(float)rand()/(RAND_MAX);
return ans;
}
main()
{
int i,sum,zero;
long N;
float ans,x,y,L,P;
srand((unsigned)time(NULL));
sum=0;
N=10000;
zero=256;/*(256,256)を原点とする*/
G_start();
G_color("white");/*白色*/
G_line(0,256,511,256);/*x 軸*/
G_line(256,0,256,511);/*y 軸*/
       for(i=0;i<N;i++){
       x=myrand();
       y=myrand();
       L=x*x+y*y;
              if(L<=1){
              G_color("blue");/*青色*/
              /*(x,y)の点を打つ。y が画面とは反対向きであることに注意*/
              G_pset((int)(x*128+zero),(int)(-y*128+zero));
              sum++;
```

```
            }
        }
ans=(float)(sum)/N;
P=ans*4;
printf("ans=%f pi=%f¥n",ans,P);
G_sleep();
G_end();
}
***ここまで***
```

モンテカルロ法による円周率プログラム 実行例: