【目次】

7JFC1121 佐藤圭一
Version.1.001 [99/01/21 22:27]

§１：理論

§１．１：凸包とは何か

凸包とは、グラフ上の点（ある点集合）の中で最も「 外側 にある点を結んで出来るものである。つまり、その線の内側にすべての点が含まれるような、いわばその点集合の最短の「境界線」と言える。

§１．２：凸包を求めるアルゴリズム

§１．２．１：包装法

ある凸包上にある点を出発点として次の凸包上の点を探す方法を考えてみると、２つの点を結んだ直線の「角度」に注目すると良い事がわかる。まず凸包上の最も左下にある点を出発点として、その点からｘ軸に平行に直線を延ばす。そして、その直線を反時計周りつまり ｘ 軸との角度を大きくしていくような方向に回転させ、角度を大きくしていく。そして最初に出会った点は、その「外側」に点がないわけなので明らかに「凸包」上の点ということになる。

さらに、そうして求めた新たな点を出発点として、同じように直線を回転させて最初にであった点が次の凸包上の点となる。ただし、直線の最初の角度はｘ 軸と平行ではなく出会ったときの角度とする。なぜなら、次の凸包上の点との角度は、必ずその前以上になり（この事は実際にグラフに点を描いていけばすぐにわかることである)、一番上の点まで行って直線の角度が１８０°を越えた場合に０°から始めると、先に「内側」の点に出会ってしまう可能性があるからである。

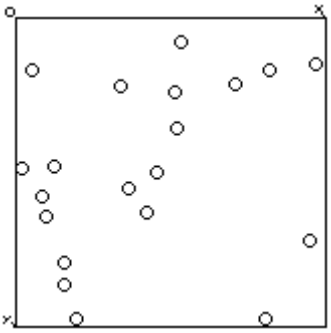
§１．２．２：Ｇｒａｈａｍ法

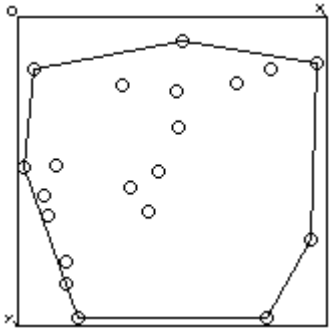
凸包に含まれる点で、最も上方にある点に注目する。注目した点より他の点を偏角順にソートし、順番に符号付き三角形を用いて演算していく方式。今回はこの方法を使った。

§２：プログラム概要

§２．１：プログラム要求仕様

Graham 法により点集合の凸包を求めるプログラムを制作する。
凸包とは、下図の様に点集合を単純な多角形で囲う事である。

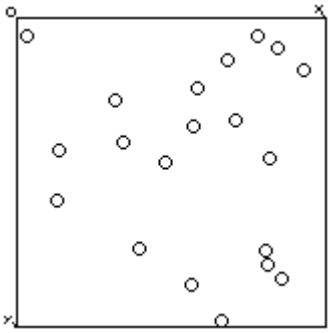




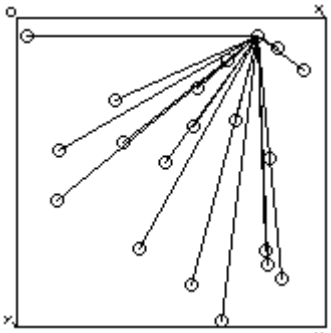
§２．２：プログラムの流れ

プログラムは下記のようなプロセスを踏む。

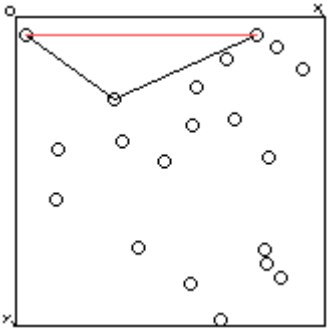
【１】
ランダムに20個、点座標を定義する。



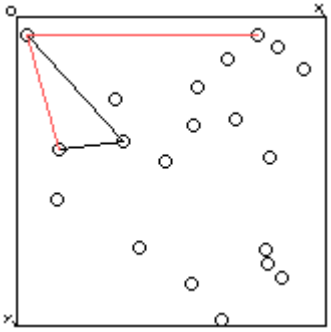
【２】
一番Ｙ軸が小さい点を見つけ、偏角順にソートする。



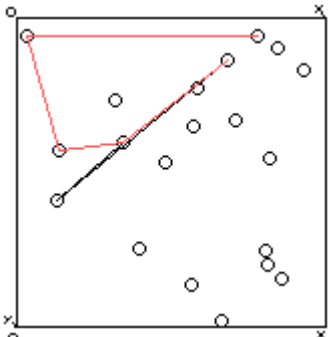
【３】
符号付き三角形演算１ステップ目



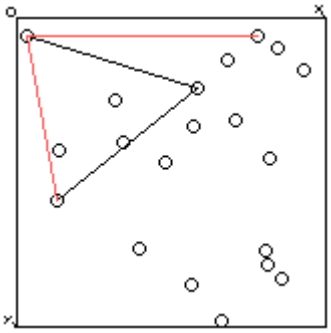
【４】
符号付き三角形演算２ステップ目



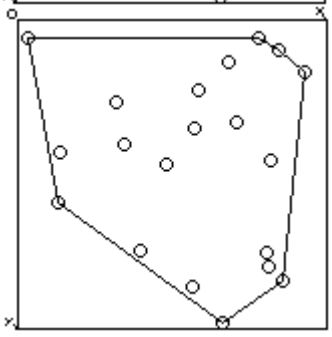
【５】
点の除去前



【６】
点の除去後



【７】
凸包完成



§２．３：工夫事項

プログラムの軸には譚教授のアルゴリズムを使用したが、よりプログラムをわかりやすくするために、少し工夫した。

- **プログラムの流れを詳細に知るために、情報出力用の手続きを用意し、現在実行しているプロセスを知ることができるようにした。**

- **大まかな Order 数がわかるように、表示プロセスに番号を振っていった。が、あくまでも目安である。**

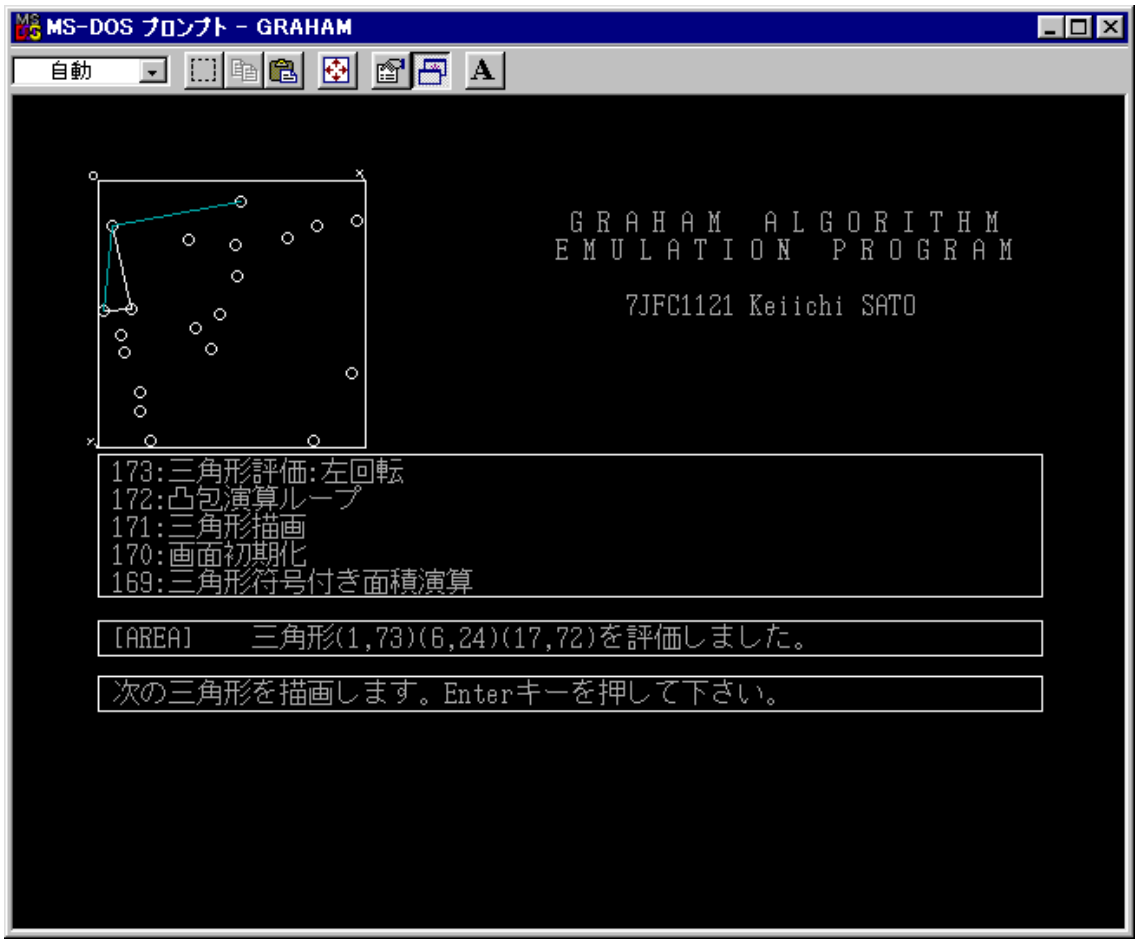
- **現在どこの手続きをしているのか、手続き名を表示するようにした。**

- **現在凸包として認識している辺と、評価中の三角形、及び三角形の評価を画面に表示する様にした。**

- **偏角順ソート画面を用意した。**

- **敢えて CRT 手続きを使わずに、すべて DOS ファンクションで処理した。よって GDI インターフェースを用意出来る DOS マシンであればどれでも実行出来る。**

§２．４：画面写真


MS-DOS プロンプト - GRAHAM
自動
GRAHAM ALGORITHM
EMULATION PROGRAM
7JFC1121 Keiichi SATO
173:三角形評価:左回転
172:凸包演算ループ
171:三角形描画
170:画面初期化
169:三角形符号付き面積演算
[AREA] 三角形(1,73)(6,24)(17,72)を評価しました。
次の三角形を描画します。Enterキーを押して下さい。

§３：結果・考察

プログラムはきちんと凸包を処理出来た。
しかし、この凸包をどのように応用するか、少々疑問である。
そこで私なりに凸包の利用価値を模索してみる事にした。
凸包は、単純な多角形を生成する。
よって、たとえば各種工事などにつかう壁紙などの料金演算に利用できそうである。
ほかにも橋脚などの３次元モデルから脆弱な部分を見つける手段としても応用が利きそうである。

§４：感想

今回は前回以前と異なり、より応用的なアルゴリズムの演習であったように思う。
よって、手段が固定されており、比較などが行えなかったのが非常に心残りである。
是非ほかのアルゴリズムも勉強していきたい。

§５：プログラムのソースコード

このプログラムは、TurboPascal5.5J / 7.0 上で開発した。
NEC PC-9821Cb 及び自作 AT 互換機にて動作確認をした。

```
program graham;(*graham Emulater By 7JFC1121 Keiichi SATO *)


(*画像定義ファイルを使用*)
uses Graph;


(*タイプの宣言*)
type point       = record
                        x,y         :       integer;
                        henkaku :       real;
                        end;
        ar_point = array[0..20] of point;
        ar_info    = array[0..5]    of string[80];
        ar_infon = array[0..20] of integer;
        str_ = string[80];


(*変数の宣言*)
var Gd,Gm,i,n,j,step : integer;
       p : ar_point;
       info : ar_info;
       infodata : str_;
       info_n : ar_infon;


(*情報出力手続き*)
procedure info_out(var info:ar_info;
                                var step:integer;
                                info_n : ar_infon;
                                data:str_);
   var i : integer;
   begin
      for i := 0 to 5 do
         info[i] := info[i+1];
      info[0] := data;
      step := step +1;
      for i := 0 to 4 do
         write('    [',14+i,';8H    [2K',step-i:3,':',info[i]);
      case info_n[0] of
         0:write('    [22;8H    [2KEnterキーを押して下さい。');
         1:write('    [22;8H    [2K演算中。暫くお待ち下さい。');
         2:write('    [22;8H    [2K次の三角形を描画します。Enter キーを押して下さい。');
         3:write('    [22;8H    [2Kプログラムを終了します。Enter キーを押して下さい。');
         4:write('    [22;8H    [2K');
      end;
      case info_n[1] of
```

```
        0:write('    [20;8H    [2K[MAIN]');
        1:write('    [20;8H    [2K[GRAHAM]');
        2:write('    [20;8H    [2K[AREA]');
        3:write('    [20;8H    [2K[SORT]');
        4:write('    [20;8H    [2K');
      end;
      case info_n[2] of
        0:write('    [20;18H乱数初期化指数[',info_n[3],']が与えられました。');
        1:begin
            write('    [20;18H三角形(',info_n[3],',',info_n[4]);
            write(')(',info_n[5],',',info_n[6],')(',info_n[7],',',info_n[8],')を評価しました。');
          end;
        2:write('    [20;18H',info_n[3]+1,'本の枝で凸包は構成されました。');
        3:write('    [20;18H');
        4:write('    [20;18H座標定義完了。');
      end;
      MoveTo(48,206) ; LineTo(48,288); MoveTo(592,288); LineTo(592,206);
      MoveTo(48,302) ; LineTo(48,322); MoveTo(592,322); LineTo(592,302);
      MoveTo(48,334) ; LineTo(48,354); MoveTo(592,354); LineTo(592,334);
    end;


(*画面初期化手続き*)
procedure graph_init(p:ar_point;j:integer) ;
    var i : integer;
    begin
      ClearDevice;
      MoveTo(48,48)   ; LineTo(48,202) ; LineTo(202,202); LineTo(202,48);
      LineTo(48,48)   ; Circle(45,45,2); MoveTo(200,45);   LineTo(197,42);
      MoveTo(197,45) ; LineTo(200,42) ; MoveTo(45,197);   LineTo(42,200);
      MoveTo(42,197) ; LineTo(43,198) ; MoveTo(202,48);   LineTo(200,46);
      MoveTo(47,202) ; LineTo(46,200) ; MoveTo(48,206);   LineTo(48,288);
      LineTo(592,288); LineTo(592,206); LineTo(48,206);   MoveTo(48,302);
      LineTo(48,322) ; LineTo(592,322); LineTo(592,302); LineTo(48,302);
      MoveTo(48,334) ; LineTo(48,354) ; LineTo(592,354); LineTo(592,334);
      LineTo(48,334) ;
      infodata := '画面初期化';
      info_out(info,step,info_n,infodata);
      if j <> 10 then
      for i := j to j+20 do
        Circle(p[i].x,p[i].y,3);
      write('    [5;40H G R A H A M     A L G O R I T H M');
      write('    [6;40HE M U L A T I O N     P R O G R A M');
      write('    [8;40H         7JFC1121 Keiichi SATO');
    end;


(*放射状に線を描く手続き*)
procedure starline(p:ar_point) ;
    var i : integer;
    begin
      infodata := '頂点検出・ソート完了';
      info_n[0] := 0;
      info_out(info,step,info_n,infodata);
```

```
    for i := 1 to 20 do
      begin
        MoveTo(p[0].x,p[0].y);
        LineTo(p[i].x,p[i].y);
      end;
    readln;
  end;


(*三角形の符号付面積を求める関数*)
function Area(q1,q2,q3:point) : integer;
  begin
    infodata := '三角形符号付き面積演算';
    info_out(info,step,info_n,infodata);
    Area := (q1.x-q3.x)*(q2.y-q3.y)+(q2.x-q3.x)*(q3.y-q1.y);
  end;


(*三角形の符号付面積の描画を行う手続き*)
procedure Area_plot(q1,q2,q3:point;p:ar_point;top:integer);
var i : integer;
  begin
    infodata := '三角形描画';
    info_out(info,step,info_n,infodata);
    graph_init(p,0);
    MoveTo(q1.x,q1.y);
    LineTo(q2.x,q2.y);
    LineTo(q3.x,q3.y);
    LineTo(q1.x,q1.y);
    for i := 1 to top do
    begin
      setcolor(3);
       MoveTo(p[i-1].x,p[i-1].y);
      LineTo(p[i].x,p[i].y);
      setcolor(15);
    end;
    if Area(q1,q2,q3) <= 0 then
      infodata := '三角形評価:右回転'
    else
      infodata := '三角形評価:左回転';
    info_n[0] := 2; info_n[1] := 2; info_n[2] := 1;
    info_n[3] := q1.x-50;   info_n[4] := q1.y-50;
    info_n[5] := q2.x-50;   info_n[6] := q2.y-50;
    info_n[7] := q3.x-50;   info_n[8] := q3.y-50;
    info_out(info,step,info_n,infodata);
    info_n[0] := 1;
    info_n[1] := 1;
    info_n[2] := 3;
    readln;
  end;


(*クイックソート手続き*)
procedure quick_sort(var p : ar_point; left,right:integer);
```

```
  var i,j,k,some : integer;
      pivot,w : point;
  begin
    info_n[1] := 3;
    infodata := 'QuickSort 実行';
    info_out(info,step,info_n,infodata);
    some   := (left + right ) div 2;
    pivot   := p[some];
    i       := left;
    j       := right;
    repeat
      begin
        while   Area(p[0],p[i],pivot) < 0 do
          i := i+1;
        while   Area(p[0],p[j],pivot) > 0 do
          j := j-1;
        if (i <= j) then
          begin
            w      := p[i];
            p[i] := p[j];
            p[j] := w;
            i := i+1;
            j := j-1
          end
      end;
    until i > j;
    if (j > left) then
      quick_sort(p,left,j);
    if (i < right) then
      quick_sort(p,i,right)
  end;


(*Graham 手続き*)
procedure GrahamScan(var p:ar_point; var j:integer);
var i,top,min,n   : integer;

  begin
    info_n[0] := 1;
    info_n[1] := 1;
    info_n[2] := 3;
    infodata := 'GrahamScan 実行';
    info_out(info,step,info_n,infodata);
    n := 20;
    min := 1;
    for i := 1 to 20 do
      begin
        if(p[i].y<p[min].y)or((p[i].y=p[min].y)and(p[i].x>p[min].x)) then
        min := i;
        infodata := '最低値検索';
        info_out(info,step,info_n,infodata);
      end;
    p[0] := p[min];
    p[min] := p[20];
```

```
    quick_sort(p,1,n-1);
    starline(p);
    info_n[1] := 1;
    top := 1;
    for i := 2 to n - 1 do
      begin
        infodata := '凸包演算ループ';
        info_out(info,step,info_n,infodata);
        Area_plot(p[top],p[top-1],p[i],p,top);
        while Area(p[top],p[top-1],p[i]) <= 0 do
          begin
            top := top - 1;
          end;
        top       := top + 1;
        p[20]    := p[top];
        p[top] := p[i];
        p[i]      := p[20];
      end;
    j := top;
  end;


(*メインルーチン*)
begin
  Gd := Detect;
  InitGraph(Gd,Gm,'a:\bgi\');
  ClearDevice;
  info_n[0] := 4;
  info_n[1] := 4;
  info_n[2] := 3;
  infodata := '    [32m    #       [0m                        Made By eucalyptus.';
  info_out(info,step,info_n,infodata);
  infodata := '    [32m ###     [0m';
  info_out(info,step,info_n,infodata);
  infodata := '    [32m#####    [0m            http://eucaly.fc.u-tokai.ac.jp/~  kei/';
  info_out(info,step,info_n,infodata);
  infodata := '    [7;31m |||     [0m';
  info_out(info,step,info_n,infodata);
  graph_init(p,10);
  write('    [20;8H    [2凸包演算プログラム By 7JFC1121  佐藤圭一(eucalyptus.)');
  write('    [22;8H    [2乱数の初期値として与える数を入力してください:');
  MoveTo(48,206); LineTo(48,288); MoveTo(592,288); LineTo(592,206);
  MoveTo(48,302); LineTo(48,322); MoveTo(592,322); LineTo(592,302);
  MoveTo(48,334); LineTo(48,354); MoveTo(592,354); LineTo(592,334);
  readln(Randseed);
  info_n[0] := 1; info_n[1] := 0;
  info_n[2] := 0; info_n[3] := Randseed;
  infodata := '座標を定義中...';
  info_out(info,step,info_n,infodata);
  for i := 1 to 20 do
    begin
      infodata := '座標定義...';
      info_out(info,step,info_n,infodata);
      p[i].x := Random(150);
```

```
            p[i].y := Random(149);
            p[i].x := p[i].x + 50;
            p[i].y := p[i].y + 51;
        end;
    graph_init(p,1);
    info_n[0] := 0;
    info_n[1] := 0;
    info_n[2] := 4;
    infodata := '座標定義完了';
    info_out(info,step,info_n,infodata);
    readln;
    GrahamScan(p,j);
    graph_init(p,0);
    MoveTo(p[0].x,p[0].y);
    for i := 1 to j do
        begin
            LineTo(p[i].x,p[i].y);
            MoveTo(p[i].x,p[i].y);
        end;
    LineTo(p[0].x,p[0].y);
    info_n[0] := 3;
    info_n[1] := 0;
    info_n[3] := j;
    info_n[2] := 2;
    infodata := '凸包完成';
    info_out(info,step,info_n,infodata);
    readln;
    CloseGraph;
end.
```

§７：添付フロッピーディスクの使用方法

添付フロッピーディスクには、下記の内容が含まれている。

graham.pas	GRAHAM 法による凸包シミュレーション
graham.exe	上記ソースを TurboPascal7.0 でコンパイルしたもの
bgi\	IBM PC/AT 互換機用 BGI ライブラリ
readme.txt	実行に必要な情報

添付プログラムを動作させるには、下記の条件を満たしている必要がある。

コンピュータ	IBM PC/AT 互換機
OS	MS-DOS PC-DOS MS-WINDOWS 日本語版
コード体系	日本語
画面モード	VGA / ANSI 対応

また、付属の BGI ライブラリは、「a:\bgi\」の下におかなければならない。