Cellular Automaton Graphics

Morio Kikuchi

Abstract：

Developing a regular polyhedron on a plane, setting discrete coordinates on the development and applying a boundary condition of regular polyhedron to it, we realize a symmetrical graphics.

1. Boundary condition

If we want to draw a symmetrical graphics on a plane, it is need that a boundary condition of a region is chosen. Boundary condition is how to connect sides of a region and I know the following three ways.

(1)no connection between sides
(2)periodic boundary condition
(3)boundary condition of regular polyhedron

The shape of a region is not always arbitrary. Figure 1 shows fundamental regions. In (1), the combination is square pixel and orthogonal coordinates and in (2), (3), the combination is hexagonal pixel and oblique coordinates. If (1) is adopted, a pixel outside region does a role of wall. We call it wall pixel. If (2) is adopted, we can understand that the same region exists periodically and innumerably on a plane.

The following are boundary conditions on the region of Figure 2.

・s1 ⇔ s3, s2 ⇔ s4：(2)
・s1 ⇔ s4, s2 ⇔ s3：(3)

If (3) is adopted, in this case, two pairs of sides being joined, a regular dihedron is made. In the region of Figure 3, a regular dihedron is made with the following correspondences.

・s1 ⇔ s6, s2 ⇔ s5, s3 ⇔ s4：(a)
・s1 ⇔ s10, s2 ⇔ s9, s3 ⇔ s8, s4 ⇔ s7, s5 ⇔ s6：(b)

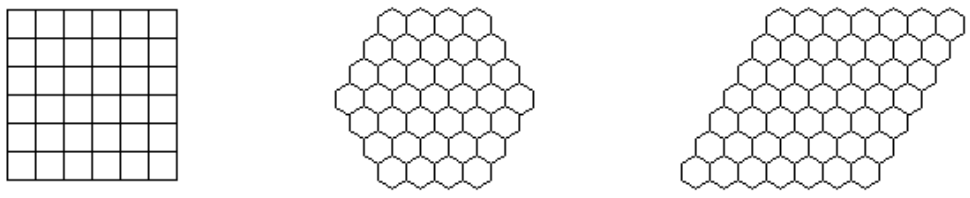


Figure 1

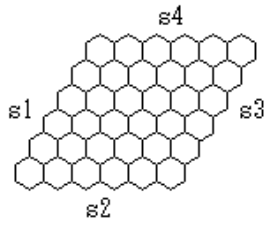
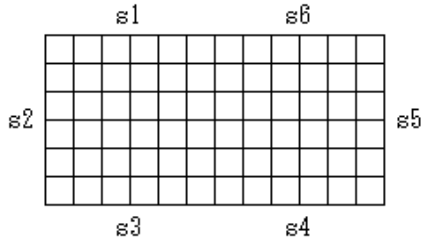
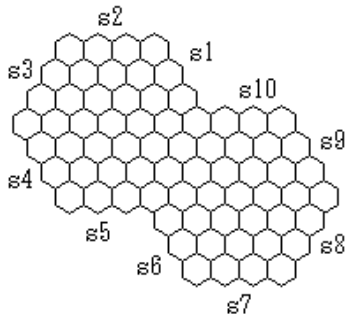


Figure 2



(a)



(b)

Figure 3

2. Symmetrical graphics

If we want to realize a symmetrical graphics, it is needed that a region is symmetrical and movements of painting point are symmetrical. The number n of pixels of the side in Figure 4-(a) is 6. The number of pixels in each regular triangle is, from the formula of arithmetic series,

$$N_{\mathrm{d}} = \frac{n}{2}(n+1) = 21$$

If two pairs of sides are joined in order to make a regular dihedron, because a pixel on a side becomes the same as the corresponding pixel on the corresponding side in the other regular triangle, the number of pixels decreases by $n-2$ on one side except two vertexes and the number of pixels decreases by 1 on one vertex. Therefore, the effective number of pixels is

$$N_2 = 2N_{\mathrm{d}} - n - 2(n-2) - 1 = n^2 - 2(n-2) - 1 = 27$$

If the number of painting points is 3 and $n = 3m + 1$, the centres of each regular triangle must be painted beforehand like Figure 4-(b).

We do a painting concretely on the case that n is 6. The coordinates of a pixel are represented with oblique coordinates in which the angle between x axis and y axis is 120 °. After arrangement of the painting number 1(a1, b1, c1) like Figure 5, the painting number 2(a2, b2, c2) is arranged as the phase difference on vector(the painting number 2 - the painting number 1) becomes 120 °. The leading painting point is the painting point 'a'. After this, the painting points are moved by the phase difference of 120 ° and Figure 6 is got up to the painting number 7. Because a pixel on a side is the same as the corresponding pixel on the corresponding side in the other regular triangle, if a pixel on a side is painted, another painting is done in the other regular triangle and it is represented with circling it. The coordinates of two corresponding pixels on the two corresponding sides in the two regular triangles are (x, y) on one regular triangle and (y, x) on the other regular triangle. Next, popping coordinates from a stack, the painting points return to the painting number 6. At the painting number 6, the neighborhood view becomes like Figure 7. If the painting point 'a' is moved to ca(a8), the other painting points are moved to ca(b8), ca(c8) of which phase difference is 120 °. The amount of change

of coordinate between searching point(the centre pixel in the neighborhood view) and ca(a8), ca(c8) in an added part in the neighborhood view exceeds the range in neighborhood view. We call the movement from searching point to a pixel in an added part in a neighborhood view jump and call a pixel in an added part in a neighborhood view jump pixel. If the painting is finished, we get Figure 8.

The each neighborhood view of Figure 7 is one on pixel which is not vertex on a side. Figure 9 is the neighborhood views on all the types of pixel on a side and the number after vertex, side shows the position in Figure 8. 1∼4 are pixel which is vertex and 5∼8 are pixel which is not vertex on a side. In side 5∼side 8, because searching point is placed next to a vertex, the two same pixels appear. However, because a pixel in a region is between them, there is no problem especially. In vertex 2, vertex 3, the two corresponding pixels in the other regular triangle can be omitted. It is assumed that the coordinates of searching point are (x, y). On a jump pixel, its coordinates are described in the figure.

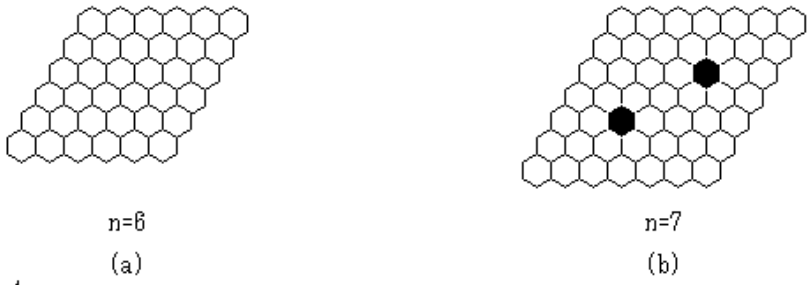


Figure 4

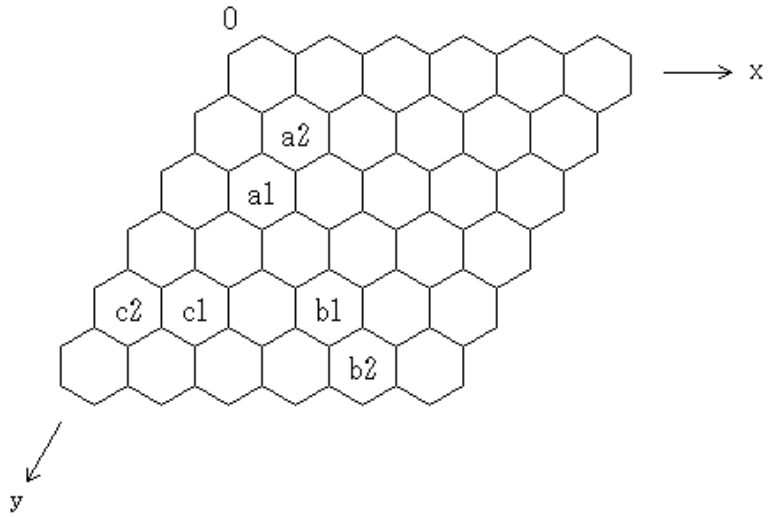


Figure 5

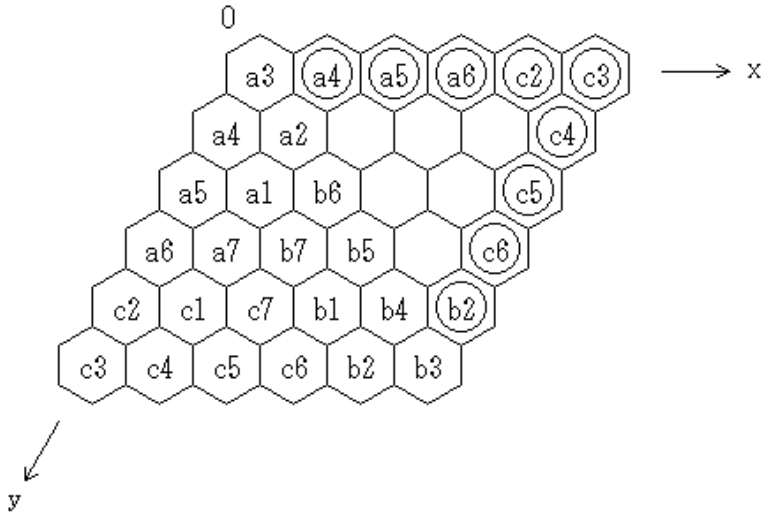
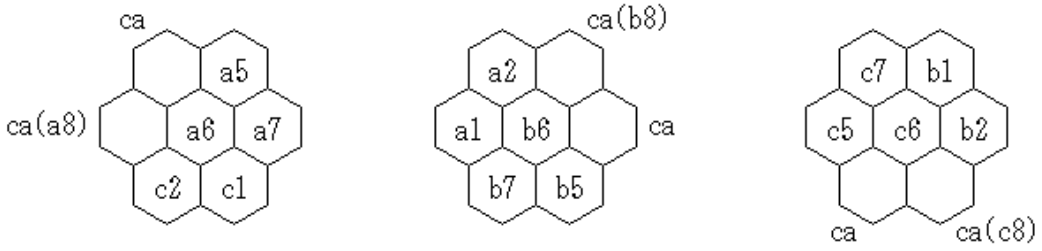


Figure 6



ca:unpainted pixel(color code:15)
ca(a8),ca(b8),ca(c8):unpainted pixel due to be painted next(color code:15)
a?,b?,c?:already painted pixel(color code:1～14)

Figure 7

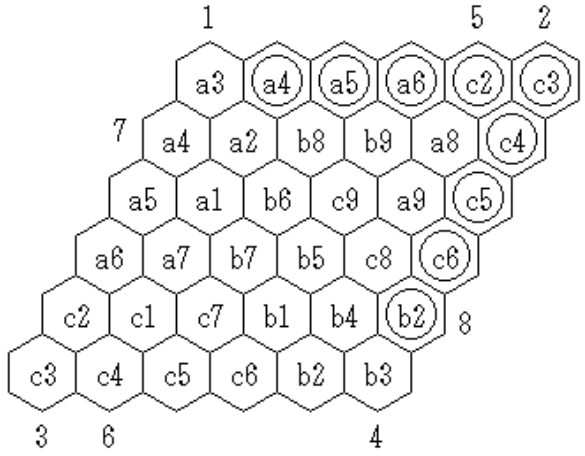


Figure 8

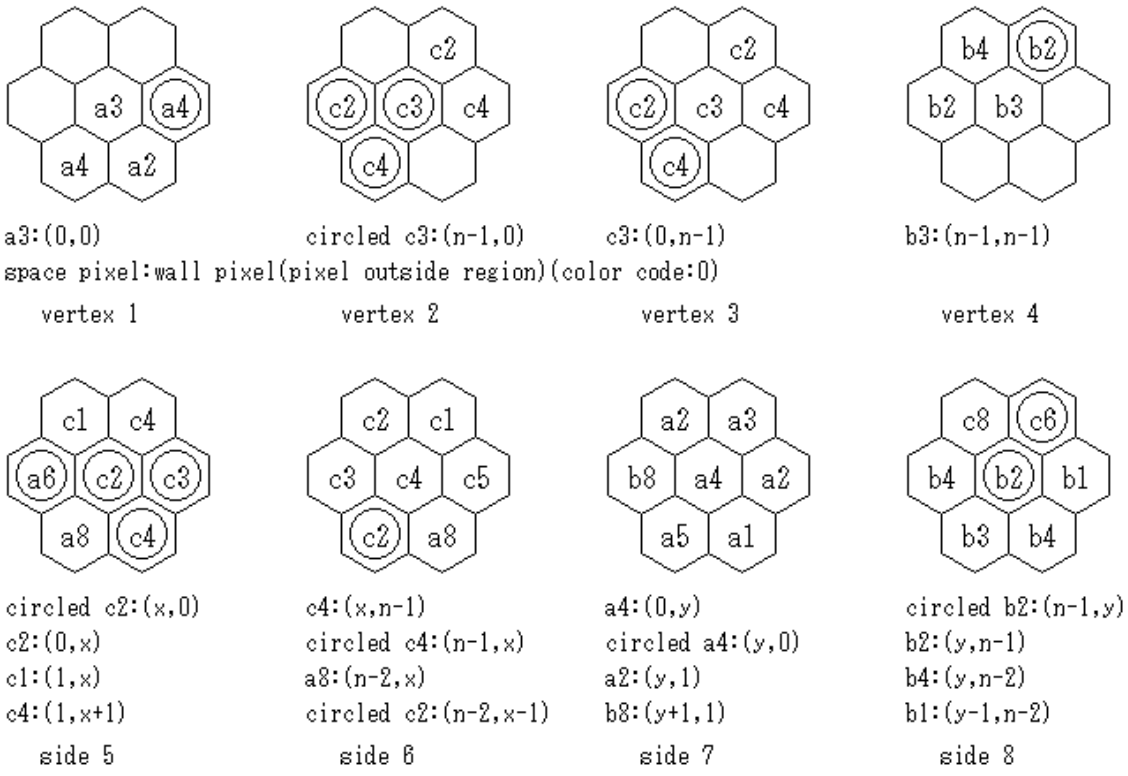


Figure 9

3. Painting algorithm

The painting at the painting number 2 is done with the algorithm of Figure 10. In the figure, S is searching point and ca is an unpainted pixel in a region namely target pixel. After this, The algorithm of Figure 11, 12 is used in order to keep symmetry at every painting number automatically. In the figures, g is a wall pixel or already painted pixel. We call g guide pixel. They are used not as they are but some devices being added to them.

It is assumed that the present searching point is S and the last searching point is S′. Like Figure 13-(a), (b), getting a signed angle between a vector by S and S′ and a vector by a target pixel ca and S, in CW, a pixel of which angle is maximum is the pixel to paint and in CCW, a pixel of which angle is minimum is the pixel to paint. We call the way angle method. If a jump pixel is chosen as a pixel to paint next, searching point jumps to a corresponding pixel(vertex or nonvertex) too.

In Figure 13, real number calculation is needed in order to angles. If each term in the algorithm of Figure 11, 12 is combined with phase difference 120 °, real number calculation is not needed. It is assumed that the painting point 'a' paints the pixel 1 in Figure 14 with the algorithm of Figure 11. The painting point 'a' is the leading painting point. The six expressions are used in if-else sentence. In if-else sentence of the painting point b, the first expression is the expression on the pixel 2 in Figure 14 and in if-else sentence of the painting point c, the first expression is the expression on the pixel 7 in Figure 14. These if-else sentences are got if if-else sentence of the painting point 'a' is modified in the state of ring. The combination 1-2-7 is used in the left regular triangle in Figure 6 and in the right regular triangle in Figure 6, the combination 1-7-2 is used. In the case that a pixel in Figure 14 is painted with the algorithm of Figure 12, too, we use the same way. We call the way shift method.

In program, there are two kinds on the timing at which a pixel is painted. In immediate painting, if a pixel to paint next is decided, it is painted immediately and in lump painting, if all pixels to paint next are decided, they are painted in a lump.

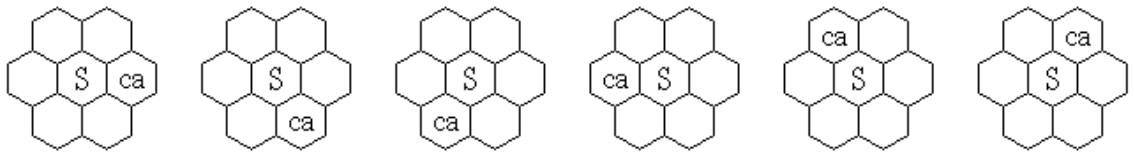


Figure 10

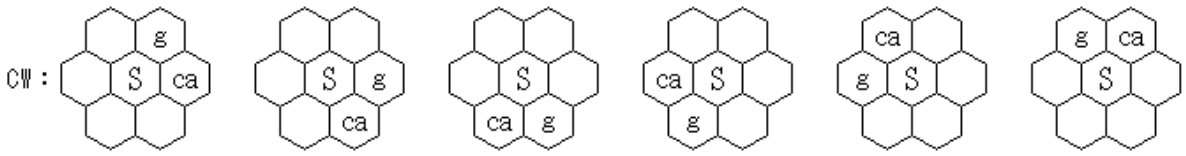


Figure 11

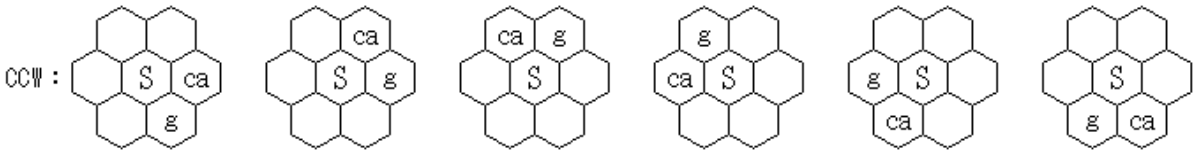


Figure 12

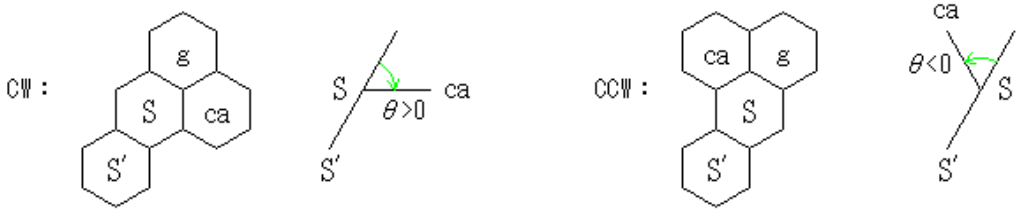


Figure 13

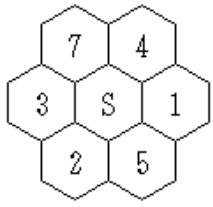
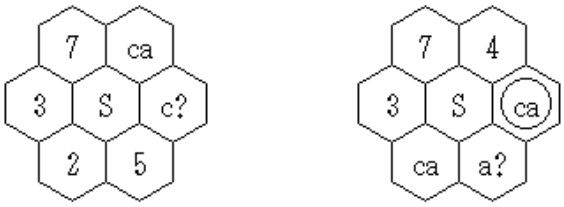


Figure 14

4. Skipping pixel

We must pay an attention to guide pixel which appears in Figure 11, 12. In Figure 9, a wall pixel(color code:0) does not appear in side 5∼side 8, however, appears in vertex 1∼vertex 4. Figure 15-(a) is a remake of vertex 3 in Figure 9 in which the two pixels is omitted. Because the pixel 5, 2, 3, 7 are wall pixel, the pixels being skipped, a? becomes guide pixel of CW. This guide pixel is common to CW, CCW. In vertex 2, the same processing as the processing in vertex 3 is done.

Figure 15-(b) is a remake of vertex 1 in Figure 9. Because the pixel 3, 7, 4 are wall pixel, the pixels being skipped and because the circled ca is the same pixel as ca, the pixel being skipped, c? becomes guide pixel of CCW. This guide pixel is common to CW, CCW. In vertex 4, the same processing as the processing in vertex 1 is done.



(a) (b)

Figure 15

5. Concrete example

Figure 16, 17 are examples of symmetrical graphics and the following are their data.

・$n = 43$

・coordinates of painting number 1：painting point 'a':(2, 4), painting point b:$(n-1-4,\ n-1-2)$, painting point c:$(2,\ n-1-2)$

・coordinates of painting number 2：painting point 'a':$\Delta y = 1$, painting point b:$\Delta x = -1; \Delta y = -1$, painting point c:$\Delta x = 1$

・choice of CW, CCW："srand(1);val=(int)((rand()/(RAND_MAX+1.))*2);"(Open Watcom C/C++ Version 1.4), val=0 ⇒ CW, val=1 ⇒ CCW

・painting algorithm：angle method(#define ALGO 0) ⇒ Figure 16；shift method(#define ALGO 1) ⇒ Figure 17

・painting timing：immediate painting

・push to stack：each time

The array which is used for painting is initialized like the following:

・target pixel：15

・centre$(n/3,\ n-(n/3+1))$, $(n-(n/3+1),\ n/3)$ of each regular triangle in Figure 4-(b)：-1

・wall pixel：0

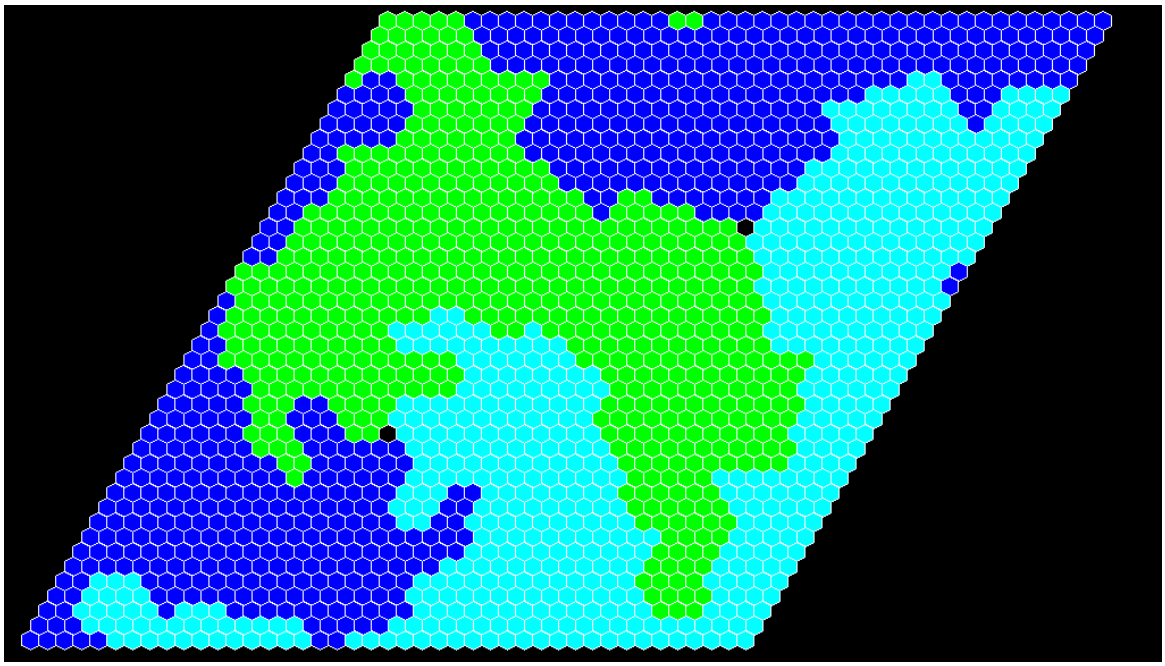


Figure 16

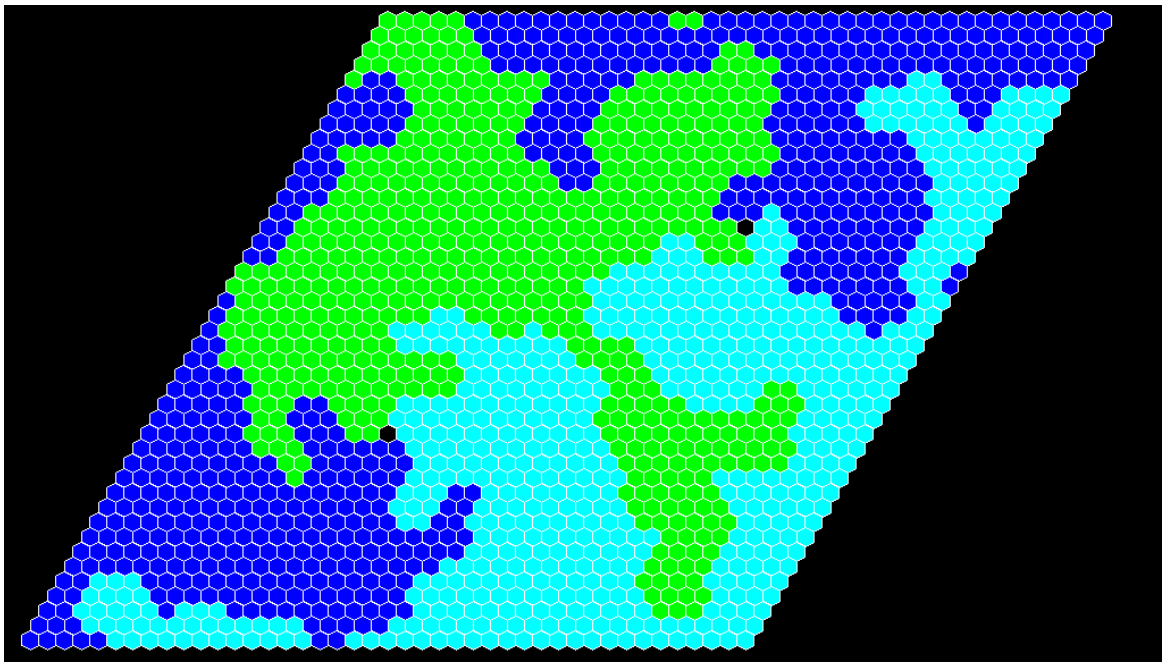


Figure 17

****

セルラーオートマトングラフィクス

菊池盛雄

アブストラクト：
　正多面体を平面上に展開し、この展開図形に離散座標を設定し、正多面体の境界条件を適用して対称なグラフィクスを実現します。

1. 境界条件

　平面上に対称なグラフィクスを描こうとすれば領域の境界条件を選択する必要があります。境界条件は領域の辺のつなぎ方であり、以下の三通りがあります。

(1) 辺をつながない
(2) 周期的境界条件
(3) 正多面体の境界条件

領域の形は任意である訳ではありません。基本的なものを図 1 に示します。(1) では正方形ピクセルと直交座標の組合せ、(2)、(3) では正六角形ピクセルと斜交座標の組合せとなります。(1) を採用すると領域の外のピクセルは壁の役目を果たします。このピクセルを壁ピクセルと称します。(2) を採用すると平面上に領域が周期的に無数にあることになります。
　以下は図 2 の領域に関する境界条件です。

・s1 ⇔ s3、s2 ⇔ s4：(2)
・s1 ⇔ s4、s2 ⇔ s3：(3)

(3) を採用するとこの場合は二組の辺が重ね合わされて正二面体ができあがります。図 3 の領域においては以下のような対応によって正二面体ができあがります。

・s1 ⇔ s6、s2 ⇔ s5、s3 ⇔ s4：(a)
・s1 ⇔ s10、s2 ⇔ s9、s3 ⇔ s8、s4 ⇔ s7、s5 ⇔ s6：(b)

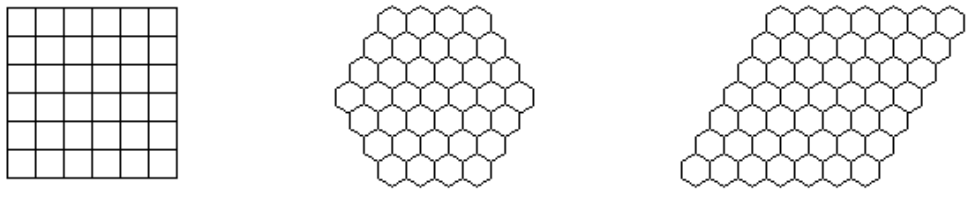


図 1

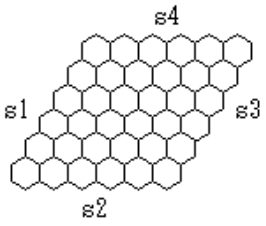


図 2

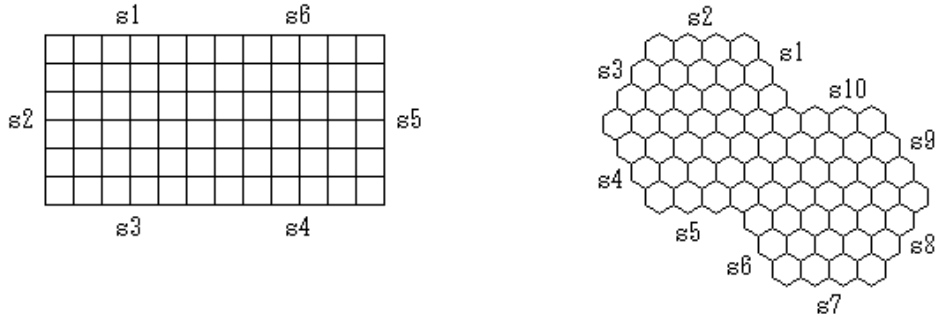


図 3

2. 対称なグラフィクス

対称なグラフィクスを実現するには領域を対称にし、塗点の動きを対称にしなければなりません。図 4-(a) の辺のピクセルの数 n は 6 です。各正三角形のピクセルの数は、等差級数の公式から

$$N_{\mathrm{d}} = \frac{n}{2}(n+1) = 21$$

正二面体を作るために辺と辺を重ねると、辺のピクセルはもう一方の正三角形の対応する辺の対応するピクセルと同一になるので、ピクセルの数は、辺に関しては頂点を除くと $n-2$ 減少し、頂点に関しては 1 減少します。したがって正二面体における有効ピクセル数は

$$N_2 = 2N_{\mathrm{d}} - n - 2(n-2) - 1 = n^2 - 2(n-2) - 1 = 27$$

塗点の数が 3 で $n = 3m + 1$ の場合は図 4-(b) のように各正三角形の中心をあらかじめ塗りつぶしておきます。

n が 6 の場合について具体的に塗りつぶしを行います。ピクセルの座標は x 軸と y 軸の角が 120 °である斜交座標で表されます。図 5 のように塗番号 1(a1、b1、c1) を配置し、ベクトル (塗番号 2-塗番号 1) の位相差が 120 °になるように塗番号 2(a2、b2、c2) を配置します。先導塗点は塗点 a とします。以後は塗点を位相差 120 °で動かし、塗番号 7 までで図 6 を得ます。辺のピクセルはもう一方の正三角形の対応する辺の対応するピクセルと同一ですから、辺のピクセルを塗りつぶすともう一方の正三角形においても塗りつぶしが行われ、これは丸で囲まれて表されます。二つの正三角形の対応する辺の対応するピクセルの座標は、一方が (x, y) ならもう一方は (y, x) です。次にスタックから座標をポップして塗番号 6 に戻ります。塗番号 6 においては近傍図は図 7 のようになります。塗点 a を ca(a8) に移せば他の塗点は位相差 120 °の ca(b8)、ca(c8) に移されます。探索点 (近傍図の中央のピクセル) と近傍図の付加部分の ca(a8)、ca(c8) における座標変化量は近傍図の範囲を越えています。探索点から近傍図の付加部分のピクセルへの移動をジャンプと称し、近傍図の付加部分のピクセルをジャンプピクセルと称します。塗りつぶしが完了すると図 8 のようになります。

図 7 の近傍図は辺の頂点ではないピクセルに関するものです。図 9 は辺のすべてのピクセルに関する近傍図であり、vertex、side の後の数字は図 8 中の位置を示しています。1 から 4 が頂点であるピクセルであり、5 から 8 が辺の頂点ではないピクセルです。side 5 から side 8 では探索点を頂点のとなりに取っ

ているので同じピクセルが二度現れます。ただし、間に領域のピクセルがあるので特に問題はありません。vertex 2、vertex 3 ではもう一方の正三角形の対応する 2 ピクセルを省略してもかまいません。ジャンプピクセルに関しては座標を図中に記してあります。

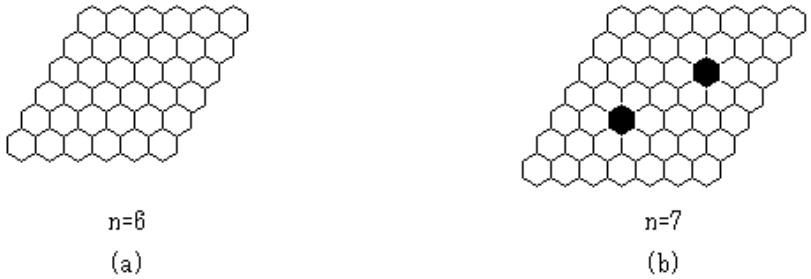


図 4

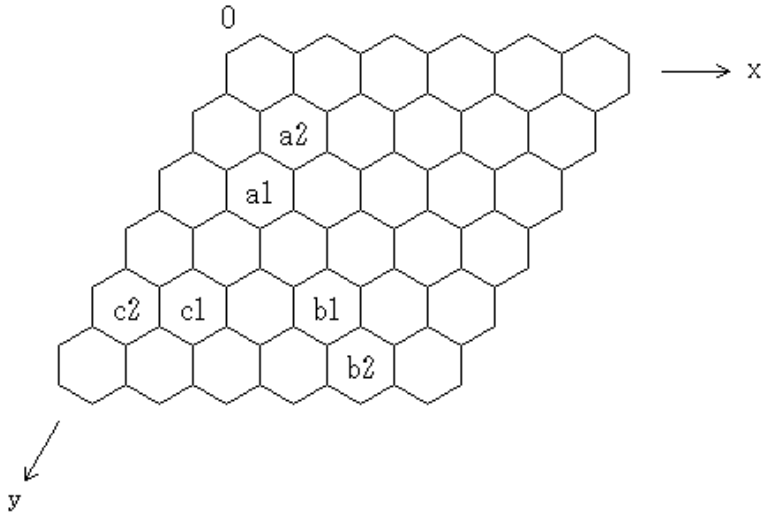


図 5

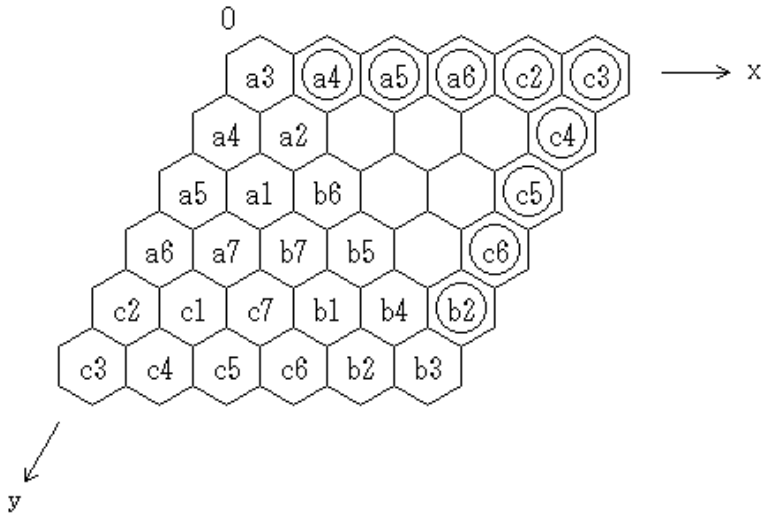
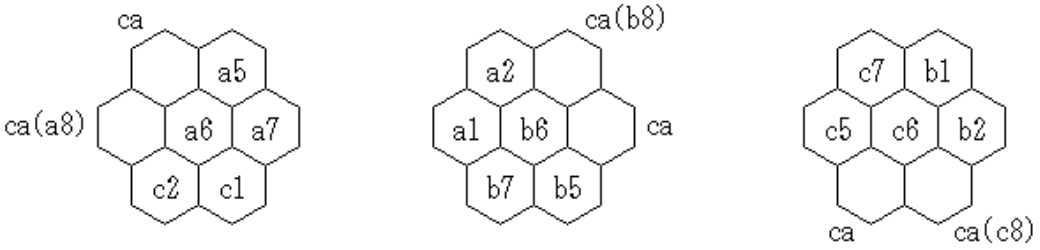


図 6



ca:unpainted pixel(color code:15)
ca(a8),ca(b8),ca(c8):unpainted pixel due to be painted next(color code:15)
a?,b?,c?:already painted pixel(color code:1～14)

図 7

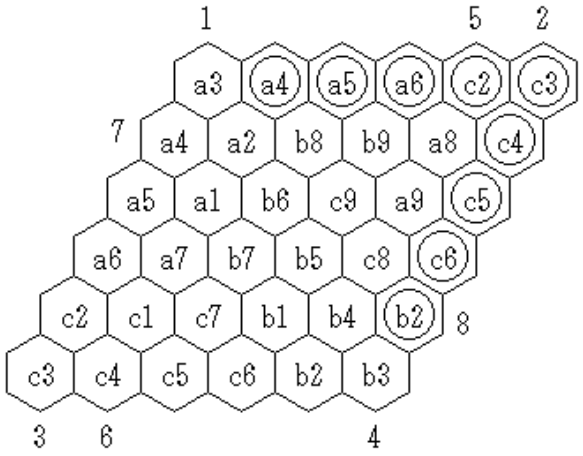
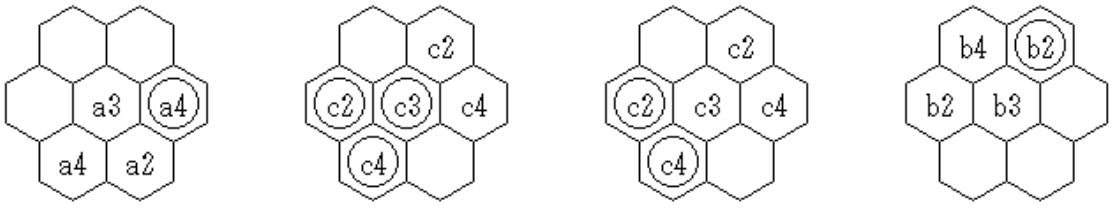
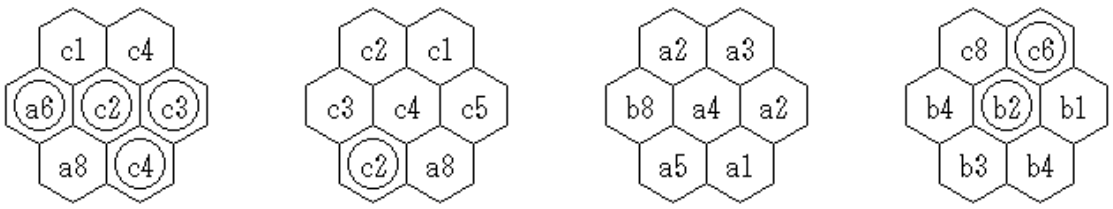


図 8



a3:(0,0)　　circled c3:(n-1,0)　　c3:(0,n-1)　　b3:(n-1,n-1)
space pixel:wall pixel(pixel outside region)(color code:0)

vertex 1　　vertex 2　　vertex 3　　vertex 4



circled c2:(x,0)　　c4:(x,n-1)　　a4:(0,y)　　circled b2:(n-1,y)
c2:(0,x)　　circled c4:(n-1,x)　　circled a4:(y,0)　　b2:(y,n-1)
c1:(1,x)　　a8:(n-2,x)　　a2:(y,1)　　b4:(y,n-2)
c4:(1,x+1)　　circled c2:(n-2,x-1)　　b8:(y+1,1)　　b1:(y-1,n-2)

side 5　　side 6　　side 7　　side 8

図 9

3. 塗りつぶしアルゴリズム

図 5 の塗番号 2 の塗りつぶしは図 10 のアルゴリズムで行います。図中において、S は探索点、ca は領

域の未塗りのピクセルすなわちターゲットピクセルです。以後は各塗番号で対称性をオートマチックに保つために図 11、12 のアルゴリズムを用います。図中において、g は壁ピクセルまたはすでに塗りつぶされたピクセルです。g をガイドピクセルと称します。これらはそのままではなく、少し手を加えます。

現在の探索点を S、S の一つ前の探索点を S′ とします。図 13-(a), (b) のように、S と S′ によるベクトルとターゲットピクセル ca と S によるベクトルのなす符号つきの角度を求め、CW ではこの角度が最大なピクセルを、CCW ではこの角度が最小なピクセルを塗りつぶすピクセルとします。このやり方を角度法と称します。次に塗りつぶすピクセルとしてジャンプピクセルが選択された場合、探索点も対応するピクセル (頂点または非頂点) にジャンプします。

図 13 では角度を求めるために実数計算が必要です。図 11、12 のアルゴリズムの各項を位相差 120 ° をつけて組み合わせれば実数計算は不要です。塗点 a が図 14 のピクセル 1 を図 11 のアルゴリズムで塗りつぶすとします。塗点 a は先導塗点です。図 11 の六つの式は if-else 文で用いられています。塗点 b の if-else 文においては最初の式を図 14 のピクセル 2 に関する式とし、塗点 c の if-else 文においては最初の式を図 14 のピクセル 7 に関する式とします。これらの if-else 文は塗点 a の if-else 文を数珠つなぎの状態にしておいて修正すれば得られます。この組合せ 1-2-7 は図 6 の左の正三角形において用い、図 6 の右の正三角形においては組合せ 1-7-2 を用います。図 14 のピクセルを図 12 のアルゴリズムで塗りつぶす場合も同様です。このやり方をシフト法と称します。

プログラムではピクセルを塗りつぶすタイミングが二種類あります。探索点で次に塗りつぶすピクセルが決まったら即時塗りつぶすやり方 (即時塗りつぶし) と、探索点で次に塗りつぶすピクセルがすべて決まったら一括して塗りつぶすやり方 (一括塗りつぶし) があります。

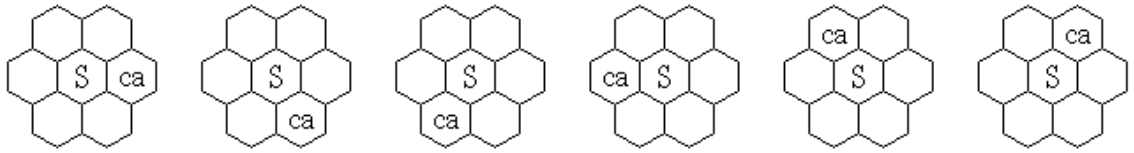


図 10

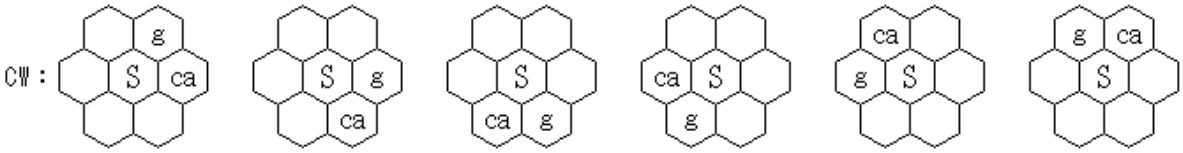


図 11

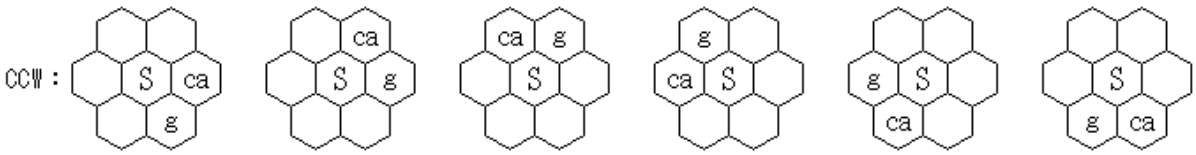


図 12

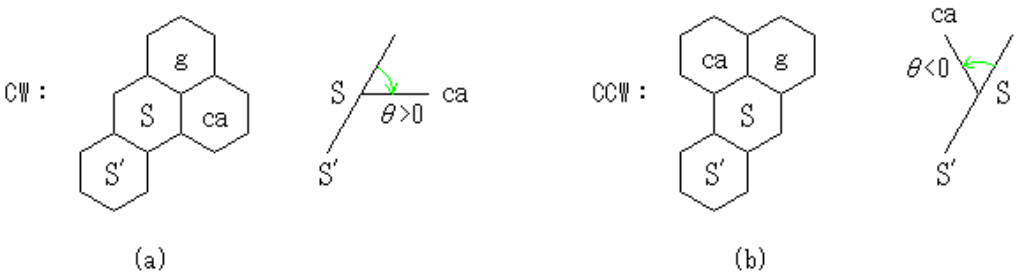


図 13

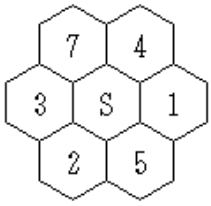


図 14

4. スキップピクセル

図 11、12 に現れるガイドピクセルには注意が必要です。図 9 において、壁ピクセル (カラーコード 0) は、side 5 から side 8 では現れませんが、vertex 1 から vertex 4 では現れます。図 9 の vertex 3 を 2 ピクセルを省略して新規に描いたのが図 15-(a) です。ピクセル 5、2、3、7 は壁ピクセルですからスキップして a?を CW のガイドピクセルとします。このガイドピクセルは CW、CCW 共通です。vertex 2 では vertex 3 と同様の処理を行います。

図 9 の vertex 1 を新規に描いたのが図 15-(b) です。ピクセル 3、7、4 は壁ピクセルですからスキップし、また丸で囲まれた ca は ca と同一のピクセルですからスキップして c?を CCW のガイドピクセルとします。このガイドピクセルは CW、CCW 共通です。vertex 4 では vertex 1 と同様の処理を行います。

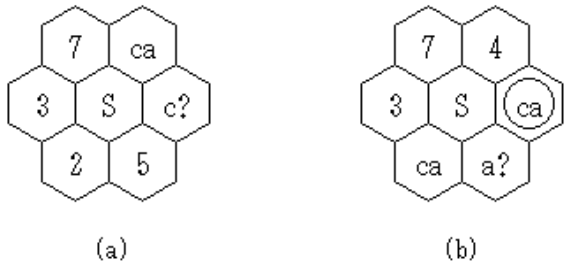


図 15

5. 具体例

図 16、17 は対称グラフィクスの例であり、以下はそのデータです。

・$n = 43$
・塗番号 1 の座標：塗点 a:(2, 4)、塗点 b:$(n-1-4,\ n-1-2)$、塗点 c:$(2,\ n-1-2)$
・塗番号 2 の座標：塗点 a:$\Delta y = 1$、塗点 b:$\Delta x = -1; \Delta y = -1$、塗点 c:$\Delta x = 1$
・CW、CCW の選択：”srand(1);val=(int)((rand()/(RAND_MAX+1.))*2);”(Open Watcom C/C++ Version 1.4), val=0 ⇒ CW, val=1 ⇒ CCW
・塗りつぶしアルゴリズム：角度法 (#define ALGO 0) ⇒ 図 16；シフト法 (#define ALGO 1) ⇒ 図 17
・塗り方：即時塗りつぶし
・スタックへの座標のプッシュ：毎回

塗りつぶしに用いられる配列は以下のように初期化します。

・ターゲットピクセル：15
・図 4-(b) の各正三角形の中心 $(n/3,\ n-(n/3+1))$、$(n-(n/3+1),\ n/3)$：-1
・壁ピクセル：0

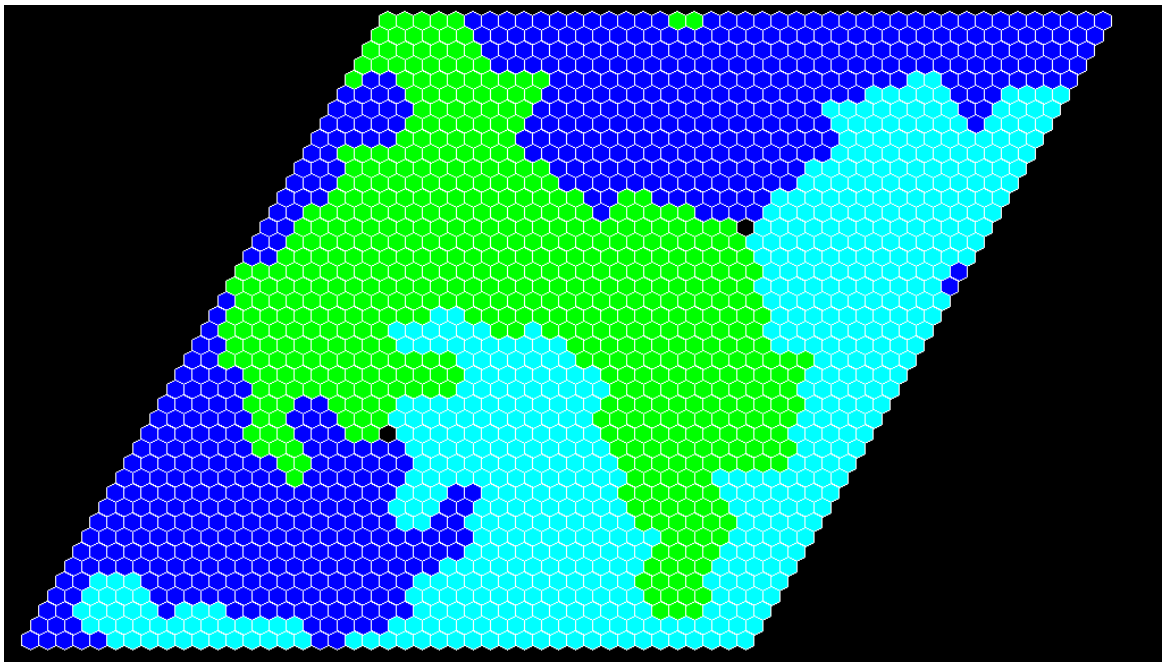


図 16

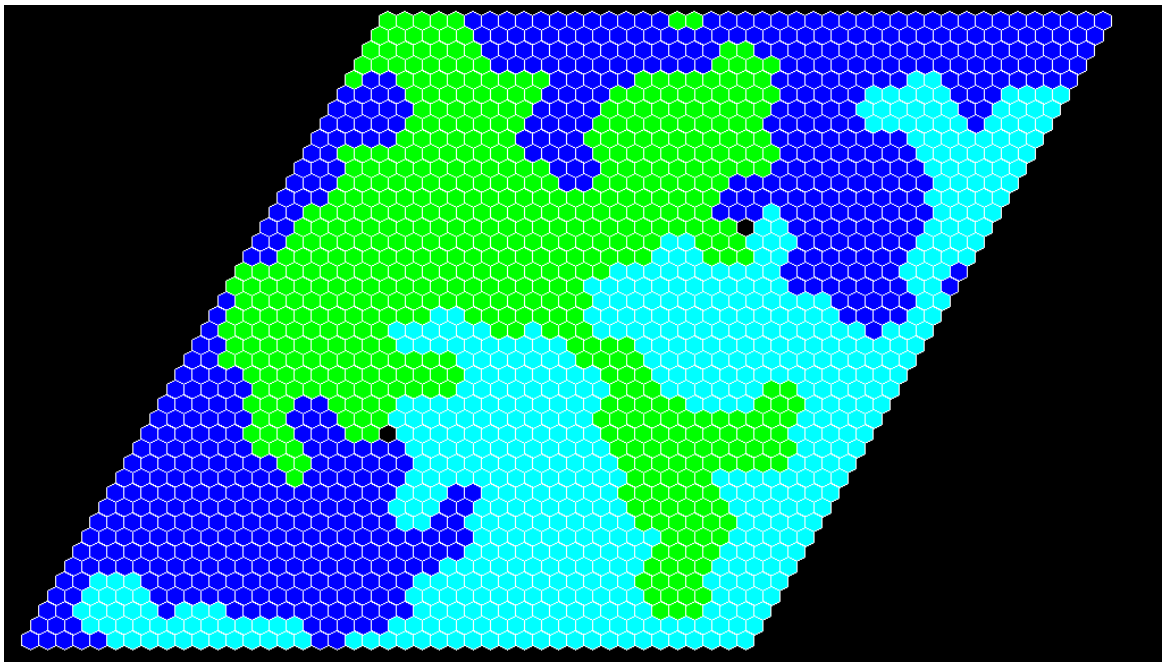


図 17

****

List 1:cag.c

```
/* t2.11                                                              */
/* 2018 Morio Kikuchi                                                 */

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define ALGO 0

#define VGACOLORS 50
#define GKS GetKeyState
#define ASIZE_MS ((1024*768)/CPMAX)  /* memory stack */
#define CPMAX 3                      /* number of painting points */
#define X0 (330)
#define Y0 (10)
#define PIXSIZE 15
#define RESO 43                      /* n */

#define ICEIL(a,b) (((a)+((b)-1))/(b))

char refill,pauseflag,fieldflag,jmpflag;
char charcode,charflag;
int Ca,C1,C2,C3,C4,C5,C7,x[8],y[8],x_[8],y_[8],cc_sum[VGACOLORS];
int X,Y,X_,Y_,Nx,Ny,Nxp,Nxm,Nyp,Nym;
int algo,combination,drn,ig;
int xt,yt,ssize,std_x,std_y,last_x,last_y;
long asize=ASIZE_MS;

/*unsigned */char **pixel;
long fp_mem[CPMAX];
double hexagon_x[6],hexagon_y[6];

char function,usflag;
unsigned char yorn;
int XRESO,YRESO,WB,DX_FRAME,DY_FRAME,DY_CAPTION;
FILE *fp;

typedef struct {int xx,yy,xx_,yy_;} ss;
ss s;ss rtn[CPMAX][ASIZE_MS];
typedef struct {
```

```
unsigned char red,green,blue;} srgb;
srgb irgb[VGACOLORS];
typedef struct {
unsigned long back_;int back,fore;} bf;
bf bfset[]={{WHITENESS,15,0},{BLACKNESS,0,15}};

HINSTANCE hinstance;
HWND hwnd;
HDC hdcdisplay,hdctmp1;
HBITMAP hbitmap1;
HPEN hpen;
HBRUSH hbrush;

void closegraph_(void),initpalette(void),BitBlt_full(void),setup(void),
     cleardevice_(char,int,int,int,int),field(void),rectangle_(int,int,int,int),
     delay_(long),beep(long),kbhit_(void),restore_3(void),initgraph_return(void),
     use_subroop(void),keydowns_f2(void),bitblt(char,int,int,int,int,int,int),
     arrayreset(void),fwrite_mem(int),fread_mem(int),putpixel_(int,int,int),
     check_rcount(void);
unsigned char subroop(void);
int initgraph_(void),setup_(void),fourfloor_fiveceil(double),random_(int),
    getpixel_(int,int,int,int),cag_r(void);
long ftell_mem(int);
double getangle(int,int);

COLORREF PALETTE(int color);
LRESULT CALLBACK wndproc_by_kbhit_(HWND,UINT,WPARAM,LPARAM);
int wndproc_filer(HWND,UINT,WPARAM,LPARAM);


int main(int argc,unsigned char **argv)
{
long mytime;

WB=1;
refill=1;
/*unlink("cpage.bin");*/

if(initgraph_()==1) return 1;

cleardevice_(1,0,0,XRESO,YRESO);
BitBlt_full();

if(setup_()==1) return 1;

if(argc>1) {time(&mytime);srand((unsigned int)mytime);}
```

```
else
srand(1);

xt=RESO-1;
yt=RESO-1;

combination=1;
drn=4;

field();
cag_r();

while(1){
check_rcount();
printf(" \n");

if(refill==0) break;
beep(50);

delay_(6000);
if(pauseflag==1) {pauseflag=0;use_subroop();}
if(refill==0) break;

/*drn=random_(6);*/

field();
cag_r();
}/**while(1)**/

closegraph_();

return 0;
}/** main **/


void ls_image(char flag,char *file,int x,int y,int dx,int dy)
{
unsigned long xsize,ysize,size;
unsigned long width,height,imagesize;
unsigned long bits,bytesPerPixel,lineSizeDW,lineSize;
HDC hdce,hdc;
HBITMAP hbitmape;
BITMAPFILEHEADER bfh;
BITMAPINFOHEADER bih;
BYTE *gdata;
FILE *fpo,*fpi;
```

```
if(flag<=3){                        /* save */
if(flag==0){
}
else if(flag==1){
}
else if(flag==2){
}
else if(flag==3){
}
else return;

if((fpo=fopen(file,"wb"))==NULL) {printf("Can't open a file.\n");return;}

width=dx;
height=dy;

bits=/*16*/24/*32*/;
bytesPerPixel=bits/8;
lineSizeDW=bytesPerPixel*width;
lineSizeDW=ICEIL(lineSizeDW,sizeof(long));
lineSize=lineSizeDW*sizeof(long);
imagesize=lineSize*height;

bfh.bfType=0x4d42;                  /* "BM" */
bfh.bfSize=54+imagesize;
bfh.bfReserved1=0;
bfh.bfOffBits=54;
bfh.bfReserved2=0;

bih.biSize=40;
bih.biWidth=width;
bih.biHeight=height;
bih.biPlanes=1;
bih.biBitCount=bits;
bih.biCompression=0;
bih.biSizeImage=imagesize;
bih.biXPelsPerMeter=0;
bih.biYPelsPerMeter=0;
bih.biClrUsed=0;
bih.biClrImportant=0;

if(flag<=1)
/*hdce=CreateCompatibleDC(hdctmp2)*/;
else if(flag==2)
hdce=CreateCompatibleDC(hdctmp1);
```

```
else{
hdc=CreateDC("DISPLAY",NULL,NULL,NULL);
hdce=CreateCompatibleDC(hdc);
}

hbitmape=CreateDIBSection(hdce,(LPBITMAPINFO)&bih,DIB_RGB_COLORS,&gdata,NULL,0);
SelectObject(hdce,hbitmape);

if(flag<=1)
/*BitBlt(hdce,0,0,dx,dy,hdctmp2,x,y,SRCCOPY)*/;
else if(flag==2)
BitBlt(hdce,0,0,dx,dy,hdctmp1,x,y,SRCCOPY);
else
BitBlt(hdce,0,0,dx,dy,hdc,x,y,SRCCOPY);

size=bih.biSizeImage;

fwrite(&bfh,14,1,fpo);
fwrite(&bih,40,1,fpo);
fwrite(gdata,size,1,fpo);

fclose(fpo);

if(flag==3) DeleteDC(hdc);
DeleteDC(hdce);
DeleteObject(hbitmape);
}
else{                                /* load */
if((fpi=fopen(file,"rb"))==NULL) {printf("Can't open the file.\n");return;}

fread(&bfh,14,1,fpi);
if(bfh.bfType!=0x4d42) {fclose(fpi);printf("Not BM.\n");return;}
fread(&bih,40,1,fpi);

fseek(fpi,bfh.bfOffBits,0);
size=bih.biSizeImage;
gdata=(BYTE *)malloc(size);
fread(gdata,size,1,fpi);

/*StretchDIBits(hdctmp2,x,y,bih.biWidth,bih.biHeight,0,0,bih.biWidth,bih.biHeight,
              gdata,(LPBITMAPINFO)&bih,DIB_RGB_COLORS,SRCCOPY);*/

fclose(fpi);
free(gdata);
}
}/** ls_image **/
```

```
void fprintf_(long v1,long v2,long v3,long v4,long v5)
{
FILE *fp;

fp=fopen("cpage.bin","ab");

fprintf(fp," %ld %ld %ld %ld %ld\n",v1,v2,v3,v4,v5);

fclose(fp);
}/** fprintf_ **/


void use_subroop(void)
{
char function_old,charflag_old;

usflag=1;

function_old=function;function=2;
charflag_old=charflag;

yorn=subroop();

function=function_old;
charflag=charflag_old;
}/** use_subroop **/


unsigned char subroop(void)
{
charflag=1;

while(1){
kbhit_();
if(charflag==0) return charcode;
}
}/** subroop **/


void keydowns_f2(void)
{
if(GKS(VK_ESCAPE)<0 || GKS(VK_PAUSE)<0) charflag=0;
else if(GKS('S')<0) {ls_image(2,"ss.bmp",0,0,XRESO,/*YRESO*/580);beep(300);}
}/** keydowns_f2 **/
```

```
void restore_in_PAINT(void)
{
ValidateRect(hwnd,NULL);

bitblt(1,0,0,XRESO,YRESO,0,0);
}/** restore_in_PAINT **/


void setup(void)
{
XRESO=1024-4;YRESO=768-24*2;
}/** setup **/


int setup_(void)
{
int i;

pixel=(/*unsigned */char **)malloc(sizeof(/*unsigned */char *)*XRESO);
if(pixel==NULL){
DeleteDC(hdctmp1);
DeleteObject(hbitmap1);
initgraph_return();return 1;}

i=0;
while(1){
pixel[i]=(/*unsigned */char *)malloc(sizeof(/*unsigned */char)*YRESO);

if(pixel[i]==NULL){
while(1){
i--;
if(i<0) break;
free(pixel[i]);
}
free(pixel);
DeleteDC(hdctmp1);
DeleteObject(hbitmap1);
initgraph_return();return 1;}

i++;
if(i==XRESO) break;
}

return 0;
```

```
}/** setup_ **/


int initgraph_(void)
{
int i,width,height;
WNDCLASS wndclass;

setup();

wndclass.hInstance    =hinstance;
wndclass.lpszClassName="CAGCLASS";
wndclass.lpszMenuName =NULL;
wndclass.lpfnWndProc  =wndproc_by_kbhit_;
wndclass.style        =0;
wndclass.hIcon        =LoadIcon(hinstance,"MYICON");
wndclass.hCursor      =LoadCursor(NULL,IDC_ARROW);
wndclass.cbClsExtra   =0;
wndclass.cbWndExtra   =0;
if(WB==0)
wndclass.hbrBackground=GetStockObject(WHITE_BRUSH);
else
wndclass.hbrBackground=GetStockObject(BLACK_BRUSH);

if(RegisterClass(&wndclass)==0) return 1;

hwnd=CreateWindow("CAGCLASS"," CAG",
                  /*WS_POPUP,*/
                  WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX,
                  0,0,XRESO+DX_FRAME,YRESO+DY_CAPTION+DY_FRAME,
                  NULL,NULL,hinstance,NULL);
if(hwnd==NULL) {MessageBox(NULL,"Memory space is not left.","CAG",MB_OK);return 1;}

SetWindowPos(hwnd,HWND_TOP,0,0,0,0,SWP_NOMOVE | SWP_NOSIZE);
ShowWindow(hwnd,SW_SHOWDEFAULT);

hdcdisplay=GetDC(hwnd);

hbitmap1=CreateCompatibleBitmap(hdcdisplay,XRESO,YRESO);
hdctmp1=CreateCompatibleDC(hdcdisplay);  /* text, dialog, menu */
SelectObject(hdctmp1,hbitmap1);
SetBkMode(hdcdisplay,TRANSPARENT);
SetBkMode(hdctmp1,TRANSPARENT);

initpalette();
```

```
SetBkColor(hdcdisplay,PALETTE(bfset[WB].back));
SetBkColor(hdctmp1,PALETTE(bfset[WB].back));

hexagon_x[0]=0.;
hexagon_x[1]=ff_fc(0.5*PIXSIZE);
hexagon_x[2]=ff_fc(1.*PIXSIZE);
hexagon_x[3]=ff_fc(1.*PIXSIZE);
hexagon_x[4]=ff_fc(0.5*PIXSIZE);
hexagon_x[5]=ff_fc(0.*PIXSIZE);

hexagon_y[0]=0.;
hexagon_y[1]=ff_fc((-sqrt(3)/6)*PIXSIZE);
hexagon_y[2]=0.;
hexagon_y[3]=ff_fc((sqrt(3)/3)*PIXSIZE);
hexagon_y[4]=ff_fc((sqrt(3)/2)*PIXSIZE);
hexagon_y[5]=ff_fc((sqrt(3)/3)*PIXSIZE);

return 0;
}/** initgraph_ **/


void initgraph_return(void)
{
/*EndPaint(hwnd,&paintstruct);*/
ReleaseDC(hwnd,hdcdisplay);
DestroyWindow(hwnd);
/*UnregisterClass("CAGCLASS",hinstance);*/

MessageBox(NULL,"Memory space is not left.","CAG",MB_OK);
}/** initgraph_return **/


void closegraph_(void)
{
int i;

i=0;
while(1){
free(pixel[i]);
i++;
if(i==XRESO) break;
}
free(pixel);

DeleteDC(hdctmp1);
DeleteObject(hbitmap1);
```

```
/*EndPaint(hwnd,&paintstruct);*/
ReleaseDC(hwnd,hdcdisplay);
DestroyWindow(hwnd);
/*UnregisterClass("CAGCLASS",hinstance);*/
}/** closegraph_ **/


void initpalette(void)
{
int i;

irgb[0].red=0;irgb[0].green=0;irgb[0].blue=0;

irgb[9].red=0;irgb[9].green=0;irgb[9].blue=255;  /* blue */
irgb[10].red=0;irgb[10].green=255;irgb[10].blue=0;  /* green */
irgb[11].red=0;irgb[11].green=255;irgb[11].blue=255;  /* cyan */
irgb[12].red=255;irgb[12].green=0;irgb[12].blue=0;  /* red */
irgb[13].red=255;irgb[13].green=0;irgb[13].blue=255;  /* magenta */
irgb[14].red=255;irgb[14].green=255;irgb[14].blue=0;  /* yellow */

irgb[15].red=255;irgb[15].green=255;irgb[15].blue=255;

for(i=7;i<9;i++){                    /* 7, 8 */
irgb[i].red=128+32*(9-i);
irgb[i].green=irgb[i].red;
irgb[i].blue=irgb[i].red;
}

for(i=16;i<20;i++){                  /* 16 -> 19 */
irgb[i].red=255-24*(20-i);
irgb[i].green=irgb[i].red;
irgb[i].blue=irgb[i].red;
}

for(i=1;i<7;i++){                    /* 1 -> 6 */
if(irgb[9+(i-1)].red==255)
irgb[i].red=irgb[9+(i-1)].red-24*1;
if(irgb[9+(i-1)].green==255)
irgb[i].green=irgb[9+(i-1)].green-24*1;
if(irgb[9+(i-1)].blue==255)
irgb[i].blue=irgb[9+(i-1)].blue-24*1;
}

for(i=20;i<26;i++){                  /* 20 -> 25 */
if(irgb[9+(i-20)].red==255)
```

```
irgb[i].red=irgb[9+(i-20)].red-24*2;
if(irgb[9+(i-20)].green==255)
irgb[i].green=irgb[9+(i-20)].green-24*2;
if(irgb[9+(i-20)].blue==255)
irgb[i].blue=irgb[9+(i-20)].blue-24*2;
}

for(i=26;i<32;i++){                 /* 26 -> 31 */
if(irgb[9+(i-26)].red==255)
irgb[i].red=irgb[9+(i-26)].red-24*3;
if(irgb[9+(i-26)].green==255)
irgb[i].green=irgb[9+(i-26)].green-24*3;
if(irgb[9+(i-26)].blue==255)
irgb[i].blue=irgb[9+(i-26)].blue-24*3;
}

for(i=32;i<38;i++){                 /* 32 -> 37 */
if(irgb[9+(i-32)].red==255)
irgb[i].red=irgb[9+(i-32)].red-24*4;
if(irgb[9+(i-32)].green==255)
irgb[i].green=irgb[9+(i-32)].green-24*4;
if(irgb[9+(i-32)].blue==255)
irgb[i].blue=irgb[9+(i-32)].blue-24*4;
}

for(i=38;i<44;i++){                 /* 38 -> 43 */
if(irgb[9+(i-38)].red==255)
irgb[i].red=irgb[9+(i-38)].red-24*5;
if(irgb[9+(i-38)].green==255)
irgb[i].green=irgb[9+(i-38)].green-24*5;
if(irgb[9+(i-38)].blue==255)
irgb[i].blue=irgb[9+(i-38)].blue-24*5;
}

for(i=44;i<50;i++){                 /* 44 -> 49 */
if(irgb[9+(i-44)].red==255)
irgb[i].red=irgb[9+(i-44)].red-24*6;
if(irgb[9+(i-44)].green==255)
irgb[i].green=irgb[9+(i-44)].green-24*6;
if(irgb[9+(i-44)].blue==255)
irgb[i].blue=irgb[9+(i-44)].blue-24*6;
}
}/** initpalette **/


void BitBlt_full(void)
```

```
{
bitblt(1,0,0,XRESO,YRESO,0,0);
}/** BitBlt_full **/


void bitblt(char flag,int x,int y,int xsize,int ysize,int x_,int y_)
{
BitBlt(hdcdisplay,x_,y_,xsize,ysize,
       hdctmp1,x,y,SRCCOPY);
}/** bitblt **/


void cleardevice_(char flag,int x,int y,int xsize,int ysize)
{
PatBlt(hdctmp1,x,y,xsize,ysize,bfset[WB].back_);
}/** cleardevice_ **/


COLORREF PALETTE(int color)
{
return RGB(irgb[color].red,irgb[color].green,irgb[color].blue);
}/** PALETTE **/


void kbhit_(void)
{
MSG msg;

if(PeekMessage(&msg,NULL,0,0,PM_REMOVE)){
TranslateMessage(&msg);
DispatchMessage(&msg);
}
}/** kbhit_ */


LRESULT CALLBACK  wndproc_by_kbhit_(HWND hwnd,UINT umsg,WPARAM wparam,LPARAM lparam)
{
if(wndproc_filer(hwnd,umsg,wparam,lparam)!=0) return 1;

return DefWindowProc(hwnd,umsg,wparam,lparam);
}/** wndproc_by_kbhit_ **/


int wndproc_filer(HWND hwnd,UINT umsg,WPARAM wparam,LPARAM lparam)
{
if(umsg==WM_KEYDOWN){
```

```
/************************** menu keydowns -> ***************************/
/************************** <- menu keydowns ***************************/

/************************** dialog keydowns -> ***************************/
/************************** <- dialog keydowns ***************************/

if(function==2){
keydowns_f2();
return 1;
}

if(usflag==1) usflag=0;

     if(GKS(VK_ESCAPE)<0 || GKS(VK_PAUSE)<0) refill=0;
else if(GKS(VK_SHIFT)<0) pauseflag=1;

return 1;
}/**else if(umsg)**/
else if(umsg==WM_SYSKEYDOWN){
}/**else if(umsg)**/
else if(umsg==WM_CLOSE){
if(function==2) charflag=0;
else refill=0;

return 1;
}/**else if(umsg)**/
else if(umsg==WM_PAINT){
restore_in_PAINT();

return 1;
}/**else if(umsg)**/
else{}

return 0;
}/** wndproc_filer **/


void delay_(long millisecond)
{
long oldtime,nowtime,dtime;
double i=CLOCKS_PER_SEC,j;

j=millisecond;
millisecond=j*(i/1000.);
oldtime=clock();
```

```
while(1){
kbhit_();
if(pauseflag==1 && refill==0) {pauseflag=0;refill=1;break;}
if(refill==0) break;

nowtime=clock();dtime=nowtime-oldtime;
if(dtime>=millisecond) break;
if(dtime<0) break;
}
}/** delay_ **/


void beep(long millisecond)
{
Beep(888,millisecond);
}/** beep **/


int fourfloor_fiveceil(double val_d)
{
int val_i,val;

val_i=floor(val_d);
val=(val_d-val_i<0.5)?val_i:val_i+1;

return val;
}/** fourfloor_fiveceil **/


int ff_fc(double val_d)
{
return fourfloor_fiveceil(val_d);
}/** ff_fc **/


void arrayreset(void)
{
int i,j;

i=0;
while(1){

j=0;
while(1){
pixel[i][j]=0;
j++;
```

```
if(j==YRESO) break;
}

i++;
if(i==XRESO) break;
}
}/** arrayreset **/


int putpixel(int nx,int ny,int pcolor)
{
int i,dx,dy;
POINT vertex[7];

if(nx<0 || ny<0 || nx>RESO-1 || ny>RESO-1) return 1;

dx=ff_fc(X0+nx*1.0*PIXSIZE-ny*0.5*PIXSIZE);
dy=ff_fc(Y0+ny*(sqrt(3)/2)*PIXSIZE);

i=0;
while(1){
vertex[i].x=hexagon_x[i]+dx;
vertex[i].y=hexagon_y[i]+dy;
i++;if(i==6) break;
}

vertex[6].x=vertex[0].x;
vertex[6].y=vertex[0].y;

if(pcolor==15)
hpen=CreatePen(PS_SOLID,1,PALETTE(9));
else
hpen=CreatePen(PS_SOLID,1,PALETTE(15));

/*if(nx==0 || ny==0 || nx==RESO-1 || ny==RESO-1) pcolor=12;*/
hbrush=CreateSolidBrush(PALETTE(pcolor));

SelectObject(hdcdisplay,hpen);
SelectObject(hdcdisplay,hbrush);

Polyline(hdcdisplay,vertex,6+1);
Polygon(hdcdisplay,vertex,6);

SelectObject(hdctmp1,hpen);
SelectObject(hdctmp1,hbrush);
```

```
Polyline(hdctmp1,vertex,6+1);
Polygon(hdctmp1,vertex,6);

DeleteObject(hbrush);
DeleteObject(hpen);

pixel[nx][ny]=pcolor;

return 0;
}/** putpixel **/


void check_rcount(void)
{
int i,j,k,m,n,dx,dy,Li,Lj;

Li=1;Lj=1;

/*999*/
for(j=0;j<Lj;j++)
for(i=0;i<Li;i++){
dx=(RESO-1)*i;
dy=(RESO-1)*j;

/* left */
if(pixel[dx][dy]!=0 && pixel[dx+RESO-1][dy+RESO-1]!=0 && pixel[dx][dy+RESO-1]!=0){
for(k=0;k<VGACOLORS;k++)
cc_sum[k]=0;

for(m=0;m<RESO;m++)
for(n=0;n<RESO;n++) {if(m>=n) cc_sum[pixel[dx+n][dy+m]]++;}

if((cc_sum[9]==cc_sum[10])&&(cc_sum[9]==cc_sum[11])){
printf(" i=%d j=%d left\n",i,j);

for(k=0;k<VGACOLORS;k++)
if(cc_sum[k]) printf(" cc%2d:%ld\n",k,cc_sum[k]);
}
}

/* right */
if(pixel[dx][dy]!=0 && pixel[dx+RESO-1][dy+RESO-1]!=0 && pixel[dx+RESO-1][dy]!=0){
for(k=0;k<VGACOLORS;k++)
cc_sum[k]=0;

for(m=0;m<RESO;m++)
```

```
for(n=0;n<RESO;n++) {if(m<=n) cc_sum[pixel[dx+n][dy+m]]++;}

if((cc_sum[9]==cc_sum[10])&&(cc_sum[9]==cc_sum[11])){
printf(" i=%d j=%d right\n",i,j);

for(k=0;k<VGACOLORS;k++)
if(cc_sum[k]) printf(" cc%2d:%ld\n",k,cc_sum[k]);
}
}
}/**for(i)**/
}/** check_rcount **/


void field_rect(int x,int y,int dx,int dy)
{
int i,j;

fieldflag=1;

for(j=y;j<y+dy;j++)
for(i=x;i<x+dx;i++){
putpixel_(i,j,15);
}

fieldflag=0;
}/** field_rect **/


void field(void)
{
field_rect(0,0,RESO,RESO);
}/** field **/


void putpixel_(int nx,int ny,int pcolor)
{
char flag;
int tmp;

putpixel(nx,ny,pcolor);
if(fieldflag) return;

/*return;*/
if(ny==nx) flag=0;                                  /* 1, 4 */
else if(nx==RESO-1 && ny==0) flag=1;
else if(nx==0 && ny==RESO-1) flag=1;
```

```
else if(nx>=1 && nx<=RESO-2 && ny==0) flag=1;       /* 5 */
else if(nx>=1 && nx<=RESO-2 && ny==RESO-1) flag=1;
else if(nx==0 && ny>=1 && ny<=RESO-2) flag=1;
else if(nx==RESO-1 && ny>=1 && ny<=RESO-2) flag=1;
else flag=-1;

if(flag==0){
}
else if(flag==1){
tmp=nx;nx=ny;ny=tmp;

putpixel(nx,ny,pcolor);
}

/*fprintf_("pp",flag,nx,ny,-1,-1);*/
}/** putpixel_ **/


void putpixel_2(int nx,int ny,int pcolor)
{
char flag;
int tmp;

if(ny==nx) flag=0;                                  /* 1, 4 */
else if(nx==RESO-1 && ny==0) flag=1;
else if(nx==0 && ny==RESO-1) flag=1;
else if(nx>=1 && nx<=RESO-2 && ny==0) flag=1;       /* 5 */
else if(nx>=1 && nx<=RESO-2 && ny==RESO-1) flag=1;
else if(nx==0 && ny>=1 && ny<=RESO-2) flag=1;
else if(nx==RESO-1 && ny>=1 && ny<=RESO-2) flag=1;
else flag=-1;

X_=nx;Y_=ny;

if(flag==0){
}
else if(flag==1){
tmp=nx;nx=ny;ny=tmp;
X_=nx;Y_=ny;
}
}/** putpixel_2 **/


int getpixel_(int x,int y,int nx,int ny)
{
int flag;
```

```
/*if(nx<0 || ny<0 || nx>RESO-1 || ny>RESO-1) return 0;
if(ny<nx-(RESO/2) || ny>nx+(RESO/2)) return 0;*/

if(x==0 && y==0) flag=1;
else if(x==RESO-1 && y==RESO-1) flag=/*2*/4;
else if(x==RESO-1 && y==0) flag=/*3*/2;
else if(x==0 && y==RESO-1) flag=/*4*/3;
else if(x>=1 && x<=RESO-2 && y==0) flag=5;
else if(x>=1 && x<=RESO-2 && y==RESO-1) flag=6;
else if(x==0 && y>=1 && y<=RESO-2) flag=7;
else if(x==RESO-1 && y>=1 && y<=RESO-2) flag=8;
else flag=0;

X=nx;Y=ny;
jmpflag=0;
/*goto end;*/

if(flag<=4){
if(nx<0 || ny<0 || nx>RESO-1 || ny>RESO-1) return 0;
}
else if(flag==5){
     if(nx==x-1 && ny==-1) {X=1;Y=x;jmpflag=1;}
else if(nx==x && ny==-1)   {X=1;Y=x+1;jmpflag=1;}
}
else if(flag==6){
     if(nx==x && ny==RESO)   {X=RESO-2;Y=x-1;jmpflag=1;}
else if(nx==x+1 && ny==RESO) {X=RESO-2;Y=x;jmpflag=1;}
}
else if(flag==7){
     if(nx==-1 && ny==y-1) {X=y;Y=1;jmpflag=1;}
else if(nx==-1 && ny==y)   {X=y+1;Y=1;jmpflag=1;}
}
else if(flag==8){
     if(nx==RESO && ny==y)   {X=y-1;Y=RESO-2;jmpflag=1;}
else if(nx==RESO && ny==y+1) {X=y;Y=RESO-2;jmpflag=1;}
}

/*if(nx<0 || ny<0 || nx>RESO-1 || ny>RESO-1)
fprintf_("gp",flag,x,y,nx,ny);*/

end:
return pixel[X][Y];
}/** getpixel_ **/
```

```
int random_(int n)
{
int val;

val=(int)((rand()/(RAND_MAX+1.))*n);

return val;
}/** random_ **/


long ftell_mem(int i)
{
return fp_mem[i];
}/** ftell_mem **/


void fwrite_mem(int i)
{
rtn[i][fp_mem[i]]=s;
fp_mem[i]++;if(fp_mem[i]>asize-1) refill=0;
}/** fwrite_mem **/


void fread_mem(int i)
{
fp_mem[i]--;if(fp_mem[i]<0) fp_mem[i]=0;
s=rtn[i][fp_mem[i]];
}/** fread_mem **/


double getangle(int next_x,int next_y)
{
char sign;
long product,p1,p2;
double val;

if(next_x==last_x && next_y==last_y) {beep(10000);refill=0;return 0;}

product=(next_x-std_x)*(std_y-last_y)-(next_y-std_y)*(std_x-last_x);
if(product==0) sign=0;
else if(product>0) sign=1;
else sign=-1;

if(sign==0) return 0;
else{
product=(next_x-std_x)*(std_x-last_x)+(next_y-std_y)*(std_y-last_y);
```

```
p1=(next_x-std_x)*(next_x-std_x)+(next_y-std_y)*(next_y-std_y);
p2=(std_x-last_x)*(std_x-last_x)+(std_y-last_y)*(std_y-last_y);
val=(double)product/(sqrt((double)p1)*sqrt((double)p2));

if(val>1) val=1;
else if(val<-1) val=-1;

return sign*acos(val);
}
}/** getangle **/


void mod_XY(int flag)
{
if(flag!=1) {putpixel_2(Nxp,Ny,0);x_[1]=X_;y_[1]=Y_;}
if(flag!=5) {putpixel_2(Nxp,Nyp,0);x_[5]=X_;y_[5]=Y_;}
if(flag!=2) {putpixel_2(Nx,Nyp,0);x_[2]=X_;y_[2]=Y_;}
if(flag!=3) {putpixel_2(Nxm,Ny,0);x_[3]=X_;y_[3]=Y_;}
if(flag!=7) {putpixel_2(Nxm,Nym,0);x_[7]=X_;y_[7]=Y_;}
if(flag!=4) {putpixel_2(Nx,Nym,0);x_[4]=X_;y_[4]=Y_;}
}/** mod_XY **/


int nv(int flag,int cc,int plc)
{
int order;

if(0) {if(cc==Ca) return 0;else return 1;}

order=1;

while(1){
if(flag==0){  /* for CW */
if(plc==1){
if(order==1){
mod_XY(5);
  if(C1==Ca) {if(x_[1]!=x[5] || y_[1]!=y[5]) return 0;}else if(C1!=0) return 1;}
else if(order==2){
  if(C4==Ca) {if(x_[4]!=x[5] || y_[4]!=y[5]) return 0;}else if(C4!=0) return 1;}
else if(order==3){
  if(C7==Ca) {if(x_[7]!=x[5] || y_[7]!=y[5]) return 0;}else if(C7!=0) return 1;}
else if(order==4){
  if(C3==Ca) {if(x_[3]!=x[5] || y_[3]!=y[5]) return 0;}else if(C3!=0) return 1;}
else if(order==5){
  if(C2==Ca) {if(x_[2]!=x[5] || y_[2]!=y[5]) return 0;}else if(C2!=0) return 1;}
else if(order==6){
```

```
  if(C5==Ca) {if(x_[5]!=x[5] || y_[5]!=y[5]) return 0;}else if(C5!=0) return 1;}
}
else if(plc==4){
if(order==1){
mod_XY(1);
  if(C4==Ca) {if(x_[4]!=x[1] || y_[4]!=y[1]) return 0;}else if(C4!=0) return 1;}
else if(order==2){
  if(C7==Ca) {if(x_[7]!=x[1] || y_[7]!=y[1]) return 0;}else if(C7!=0) return 1;}
else if(order==3){
  if(C3==Ca) {if(x_[3]!=x[1] || y_[3]!=y[1]) return 0;}else if(C3!=0) return 1;}
else if(order==4){
  if(C2==Ca) {if(x_[2]!=x[1] || y_[2]!=y[1]) return 0;}else if(C2!=0) return 1;}
else if(order==5){
  if(C5==Ca) {if(x_[5]!=x[1] || y_[5]!=y[1]) return 0;}else if(C5!=0) return 1;}
else if(order==6){
  if(C1==Ca) {if(x_[1]!=x[1] || y_[1]!=y[1]) return 0;}else if(C1!=0) return 1;}
}
else if(plc==7){
if(order==1){
mod_XY(4);
  if(C7==Ca) {if(x_[7]!=x[4] || y_[7]!=y[4]) return 0;}else if(C7!=0) return 1;}
else if(order==2){
  if(C3==Ca) {if(x_[3]!=x[4] || y_[3]!=y[4]) return 0;}else if(C3!=0) return 1;}
else if(order==3){
  if(C2==Ca) {if(x_[2]!=x[4] || y_[2]!=y[4]) return 0;}else if(C2!=0) return 1;}
else if(order==4){
  if(C5==Ca) {if(x_[5]!=x[4] || y_[5]!=y[4]) return 0;}else if(C5!=0) return 1;}
else if(order==5){
  if(C1==Ca) {if(x_[1]!=x[4] || y_[1]!=y[4]) return 0;}else if(C1!=0) return 1;}
else if(order==6){
  if(C4==Ca) {if(x_[4]!=x[4] || y_[4]!=y[4]) return 0;}else if(C4!=0) return 1;}
}
else if(plc==3){
if(order==1){
mod_XY(7);
  if(C3==Ca) {if(x_[3]!=x[7] || y_[3]!=y[7]) return 0;}else if(C3!=0) return 1;}
else if(order==2){
  if(C2==Ca) {if(x_[2]!=x[7] || y_[2]!=y[7]) return 0;}else if(C2!=0) return 1;}
else if(order==3){
  if(C5==Ca) {if(x_[5]!=x[7] || y_[5]!=y[7]) return 0;}else if(C5!=0) return 1;}
else if(order==4){
  if(C1==Ca) {if(x_[1]!=x[7] || y_[1]!=y[7]) return 0;}else if(C1!=0) return 1;}
else if(order==5){
  if(C4==Ca) {if(x_[4]!=x[7] || y_[4]!=y[7]) return 0;}else if(C4!=0) return 1;}
else if(order==6){
  if(C7==Ca) {if(x_[7]!=x[7] || y_[7]!=y[7]) return 0;}else if(C7!=0) return 1;}
```

```
}
else if(plc==2){
if(order==1){
mod_XY(3);
  if(C2==Ca) {if(x_[2]!=x[3] || y_[2]!=y[3]) return 0;}else if(C2!=0) return 1;}
else if(order==2){
  if(C5==Ca) {if(x_[5]!=x[3] || y_[5]!=y[3]) return 0;}else if(C5!=0) return 1;}
else if(order==3){
  if(C1==Ca) {if(x_[1]!=x[3] || y_[1]!=y[3]) return 0;}else if(C1!=0) return 1;}
else if(order==4){
  if(C4==Ca) {if(x_[4]!=x[3] || y_[4]!=y[3]) return 0;}else if(C4!=0) return 1;}
else if(order==5){
  if(C7==Ca) {if(x_[7]!=x[3] || y_[7]!=y[3]) return 0;}else if(C7!=0) return 1;}
else if(order==6){
  if(C3==Ca) {if(x_[3]!=x[3] || y_[3]!=y[3]) return 0;}else if(C3!=0) return 1;}
}
else if(plc==5){
if(order==1){
mod_XY(2);
  if(C5==Ca) {if(x_[5]!=x[2] || y_[5]!=y[2]) return 0;}else if(C5!=0) return 1;}
else if(order==2){
  if(C1==Ca) {if(x_[1]!=x[2] || y_[1]!=y[2]) return 0;}else if(C1!=0) return 1;}
else if(order==3){
  if(C4==Ca) {if(x_[4]!=x[2] || y_[4]!=y[2]) return 0;}else if(C4!=0) return 1;}
else if(order==4){
  if(C7==Ca) {if(x_[7]!=x[2] || y_[7]!=y[2]) return 0;}else if(C7!=0) return 1;}
else if(order==5){
  if(C3==Ca) {if(x_[3]!=x[2] || y_[3]!=y[2]) return 0;}else if(C3!=0) return 1;}
else if(order==6){
  if(C2==Ca) {if(x_[2]!=x[2] || y_[2]!=y[2]) return 0;}else if(C2!=0) return 1;}
}
}/**if(flag)**/
else{  /* for CCW */
if(plc==1){
if(order==1){
mod_XY(4);
  if(C1==Ca) {if(x_[1]!=x[4] || y_[1]!=y[4]) return 0;}else if(C1!=0) return 1;}
else if(order==2){
  if(C5==Ca) {if(x_[5]!=x[4] || y_[5]!=y[4]) return 0;}else if(C5!=0) return 1;}
else if(order==3){
  if(C2==Ca) {if(x_[2]!=x[4] || y_[2]!=y[4]) return 0;}else if(C2!=0) return 1;}
else if(order==4){
  if(C3==Ca) {if(x_[3]!=x[4] || y_[3]!=y[4]) return 0;}else if(C3!=0) return 1;}
else if(order==5){
  if(C7==Ca) {if(x_[7]!=x[4] || y_[7]!=y[4]) return 0;}else if(C7!=0) return 1;}
else if(order==6){
```

```
  if(C4==Ca) {if(x_[4]!=x[4] || y_[4]!=y[4]) return 0;}else if(C4!=0) return 1;}
}
else if(plc==5){
if(order==1){
mod_XY(1);
  if(C5==Ca) {if(x_[5]!=x[1] || y_[5]!=y[1]) return 0;}else if(C5!=0) return 1;}
else if(order==2){
  if(C2==Ca) {if(x_[2]!=x[1] || y_[2]!=y[1]) return 0;}else if(C2!=0) return 1;}
else if(order==3){
  if(C3==Ca) {if(x_[3]!=x[1] || y_[3]!=y[1]) return 0;}else if(C3!=0) return 1;}
else if(order==4){
  if(C7==Ca) {if(x_[7]!=x[1] || y_[7]!=y[1]) return 0;}else if(C7!=0) return 1;}
else if(order==5){
  if(C4==Ca) {if(x_[4]!=x[1] || y_[4]!=y[1]) return 0;}else if(C4!=0) return 1;}
else if(order==6){
  if(C1==Ca) {if(x_[1]!=x[1] || y_[1]!=y[1]) return 0;}else if(C1!=0) return 1;}
}
else if(plc==2){
if(order==1){
mod_XY(5);
  if(C2==Ca) {if(x_[2]!=x[5] || y_[2]!=y[5]) return 0;}else if(C2!=0) return 1;}
else if(order==2){
  if(C3==Ca) {if(x_[3]!=x[5] || y_[3]!=y[5]) return 0;}else if(C3!=0) return 1;}
else if(order==3){
  if(C7==Ca) {if(x_[7]!=x[5] || y_[7]!=y[5]) return 0;}else if(C7!=0) return 1;}
else if(order==4){
  if(C4==Ca) {if(x_[4]!=x[5] || y_[4]!=y[5]) return 0;}else if(C4!=0) return 1;}
else if(order==5){
  if(C1==Ca) {if(x_[1]!=x[5] || y_[1]!=y[5]) return 0;}else if(C1!=0) return 1;}
else if(order==6){
  if(C5==Ca) {if(x_[5]!=x[5] || y_[5]!=y[5]) return 0;}else if(C5!=0) return 1;}
}
else if(plc==3){
if(order==1){
mod_XY(2);
  if(C3==Ca) {if(x_[3]!=x[2] || y_[3]!=y[2]) return 0;}else if(C3!=0) return 1;}
else if(order==2){
  if(C7==Ca) {if(x_[7]!=x[2] || y_[7]!=y[2]) return 0;}else if(C7!=0) return 1;}
else if(order==3){
  if(C4==Ca) {if(x_[4]!=x[2] || y_[4]!=y[2]) return 0;}else if(C4!=0) return 1;}
else if(order==4){
  if(C1==Ca) {if(x_[1]!=x[2] || y_[1]!=y[2]) return 0;}else if(C1!=0) return 1;}
else if(order==5){
  if(C5==Ca) {if(x_[5]!=x[2] || y_[5]!=y[2]) return 0;}else if(C5!=0) return 1;}
else if(order==6){
  if(C2==Ca) {if(x_[2]!=x[2] || y_[2]!=y[2]) return 0;}else if(C2!=0) return 1;}
```

```
}
else if(plc==7){
if(order==1){
mod_XY(3);
  if(C7==Ca) {if(x_[7]!=x[3] || y_[7]!=y[3]) return 0;}else if(C7!=0) return 1;}
else if(order==2){
  if(C4==Ca) {if(x_[4]!=x[3] || y_[4]!=y[3]) return 0;}else if(C4!=0) return 1;}
else if(order==3){
  if(C1==Ca) {if(x_[1]!=x[3] || y_[1]!=y[3]) return 0;}else if(C1!=0) return 1;}
else if(order==4){
  if(C5==Ca) {if(x_[5]!=x[3] || y_[5]!=y[3]) return 0;}else if(C5!=0) return 1;}
else if(order==5){
  if(C2==Ca) {if(x_[2]!=x[3] || y_[2]!=y[3]) return 0;}else if(C2!=0) return 1;}
else if(order==6){
  if(C3==Ca) {if(x_[3]!=x[3] || y_[3]!=y[3]) return 0;}else if(C3!=0) return 1;}
}
else if(plc==4){
if(order==1){
mod_XY(7);
  if(C4==Ca) {if(x_[4]!=x[7] || y_[4]!=y[7]) return 0;}else if(C4!=0) return 1;}
else if(order==2){
  if(C1==Ca) {if(x_[1]!=x[7] || y_[1]!=y[7]) return 0;}else if(C1!=0) return 1;}
else if(order==3){
  if(C5==Ca) {if(x_[5]!=x[7] || y_[5]!=y[7]) return 0;}else if(C5!=0) return 1;}
else if(order==4){
  if(C2==Ca) {if(x_[2]!=x[7] || y_[2]!=y[7]) return 0;}else if(C2!=0) return 1;}
else if(order==5){
  if(C3==Ca) {if(x_[3]!=x[7] || y_[3]!=y[7]) return 0;}else if(C3!=0) return 1;}
else if(order==6){
  if(C7==Ca) {if(x_[7]!=x[7] || y_[7]!=y[7]) return 0;}else if(C7!=0) return 1;}
}
}/**else(flag)**/

order++;if(order==6) return 0;
}

return -1;
}/** nv **/


int where_cp(int *nx,int *ny)
{
int i,flag_l,flag_r;

flag_l=flag_r=0;
```

```
i=0;
while(1){
if(ny[i]>=nx[i])      flag_l++;  /* left */
else if(ny[i]<=nx[i]) flag_r++;  /* right */

i++;if(i==CPMAX) break;
}

if(flag_l>flag_r) return 0;else return 1;
}/** where_cp **/


long get_len(int x1,int y1,int x2,int y2)
{
int dx,dy;

dx=x2-x1;
dy=y2-y1;

return /*sqrt*/(dx*dx+dy*dy-dx*dy);
}/** get_len **/


int check_len(int *nx,int *ny)
{
int i,j,k,m,dx,dy,lr,Li,Lj;
long len[3];

Li=1;Lj=1;

/*999*/
for(j=0;j<Lj;j++)
for(i=0;i<Li;i++){
dx=(RESO-1)*i;
dy=(RESO-1)*j;

/* left */
if(pixel[dx][dy]!=0 && pixel[dx+RESO-1][dy+RESO-1]!=0 && pixel[dx][dy+RESO-1]!=0){
for(k=0;k<CPMAX;k++){
if(nx[k]>=dx && ny[k]<=dy+RESO-1 && ny[k]>=nx[k]-dx+dy) ;
else goto right;
}

lr=0;
goto next;
}
```

```
/* right */
right:
if(pixel[dx][dy]!=0 && pixel[dx+RESO-1][dy+RESO-1]!=0 && pixel[dx+RESO-1][dy]!=0){
for(k=0;k<CPMAX;k++){
if(nx[k]<=dx+RESO-1 && ny[k]>=dy && ny[k]<=nx[k]-dx+dy) ;
else goto next_for;
}

lr=1;
goto next;
}

next_for:
;
}/**for(i)**/

return 0;

next:
for(k=0;k<CPMAX;k++){
if(k<CPMAX-1) m=k+1;else m=0;
len[k]=get_len(nx[k],ny[k],nx[m],ny[m]);
}

if(len[0]==len[1] && len[1]==len[2]){
/*printf(" i=%d j=%d\n",i,j);*/
if(RESO%3==1){
if(lr==0) {dx=RESO/3+(RESO-1)*i;          dy=RESO-(RESO/3+1)+(RESO-1)*j;}
else      {dx=RESO-(RESO/3+1)+(RESO-1)*i;dy=RESO/3+(RESO-1)*j;}

for(k=0;k<CPMAX;k++)
len[k]=get_len(dx,dy,nx[k],ny[k]);

if(len[0]==len[1] && len[1]==len[2]) return 1;
else return 2;
}/**if(RESO%3)**/
else return 1;
}/**if(len[]==len[])**/
else return 2;
}/** check_len **/


int cag_r(void)
{
int i,flag_us,flag_lr;
```

```
int flag_[CPMAX],flag_pp[CPMAX],acolor[CPMAX];
int nx[CPMAX],ny[CPMAX],nx_[CPMAX],ny_[CPMAX],nax[CPMAX],nay[CPMAX];
int XDP,YDP,nx_old[CPMAX],ny_old[CPMAX];
int cp,ssize;
int ca,c1,c2,c3,c4,c5,c7;
int nxp,nxm,nyp,nym;
int jmp[8],jp,tmp;
int fw[2][6][CPMAX],al0,ali;
double val,angle;

fw[0][0][1]=2;fw[0][0][2]=4;  /* 1-2-7 */
fw[0][1][1]=3;fw[0][1][2]=5;  /* 5-3-4 */
fw[0][2][1]=4;fw[0][2][2]=0;  /* 2-7-1 */
fw[0][3][1]=5;fw[0][3][2]=1;  /* 3-4-5 */
fw[0][4][1]=0;fw[0][4][2]=2;  /* 7-1-2 */
fw[0][5][1]=1;fw[0][5][2]=3;  /* 4-5-3 */

fw[1][0][1]=4;fw[1][0][2]=2;  /* 1-7-2 */
fw[1][1][1]=5;fw[1][1][2]=3;  /* 5-4-3 */
fw[1][2][1]=0;fw[1][2][2]=4;  /* 2-1-7 */
fw[1][3][1]=1;fw[1][3][2]=5;  /* 3-5-4 */
fw[1][4][1]=2;fw[1][4][2]=0;  /* 7-2-1 */
fw[1][5][1]=3;fw[1][5][2]=1;  /* 4-3-5 */

/*putpixel_2(1,0,0);printf(" :%d %d\n",X_,Y_);
putpixel_2(0,1,0);printf(" :%d %d\n",X_,Y_);*/
if(0){
pixel[0][0]=-1;pixel[RESO-1][0]=-1;
pixel[0][RESO-1]=-1;pixel[RESO-1][RESO-1]=-1;
}
if(RESO%3==1){
XDP=RESO/3;YDP=RESO-(RESO/3+1);pixel[XDP][YDP]=-1;
XDP=RESO-(RESO/3+1);YDP=RESO/3;pixel[XDP][YDP]=-1;
}

ssize=sizeof(ss);
cp=CPMAX;

acolor[0]=9;acolor[1]=10;acolor[2]=11;

for(i=0;i<CPMAX;i++){
flag_[i]=1;
}

for(i=0;i<CPMAX;i++)
fp_mem[i]=0;
```

```
ca=15;
Ca=ca;

nax[0]=2     ;nay[0]=4;
if(combination==/*0*/1){
nax[1]=RESO-5;nay[1]=RESO-3;
nax[2]=2     ;nay[2]=RESO-3;
}
else{
nax[2]=RESO-5;nay[2]=RESO-3;
nax[1]=2     ;nay[1]=RESO-3;
}

i=0;
while(1){
if(flag_[i]){                        /* CP_? */
ig=i;

nx[i]=nax[i];ny[i]=nay[i];
putpixel_(nx[i],ny[i],acolor[i]);
}/**if(flag_[i])**/

i++;if(i==CPMAX) break;
}/**while(1)**/

i=0;
while(1){
if(flag_[i]){                        /* CP_? */
nx_[i]=nax[i];ny_[i]=nay[i];
ig=i;

if(combination==0){
/* CW */
}/**if(combination)**/
else{
/* CCW */
if(drn==4) {/* 217 */
     if(i%3==0) {nay[i]++;}
else if(i%3==1) {nax[i]--;nay[i]--;}
else if(i%3==2) {nax[i]++;}
}
}/**else(combination)**/

nx[i]=nax[i];ny[i]=nay[i];
```

```
putpixel_(nx[i],ny[i],acolor[i]);
}/**if(flag_[i])**/

i++;if(i==CPMAX) break;
}/**while(1)**/

/***************************** while(cp) -> ****************************/

flag_us=0;

while(cp){                  /* 2pie/3 */
kbhit_();
if(refill==0) break;

algo=random_(2);
flag_lr=where_cp(nx,ny);

for(i=0;i<CPMAX;i++){
nx_old[i]=nx[i];
ny_old[i]=ny[i];
}

i=0;
while(1){

if(flag_[i]){                       /* CP_? */
ig=i;

nxp=nx[i]+1;nyp=ny[i]+1;nxm=nx[i]-1;nym=ny[i]-1;
c1=getpixel_(nx[i],ny[i],nxp,ny[i]);
x[1]=X;y[1]=Y;jmp[1]=jmpflag;
c2=getpixel_(nx[i],ny[i],nx[i],nyp);
x[2]=X;y[2]=Y;jmp[2]=jmpflag;
c3=getpixel_(nx[i],ny[i],nxm,ny[i]);
x[3]=X;y[3]=Y;jmp[3]=jmpflag;
c4=getpixel_(nx[i],ny[i],nx[i],nym);
x[4]=X;y[4]=Y;jmp[4]=jmpflag;
c5=getpixel_(nx[i],ny[i],nxp,nyp);
x[5]=X;y[5]=Y;jmp[5]=jmpflag;
c7=getpixel_(nx[i],ny[i],nxm,nym);
x[7]=X;y[7]=Y;jmp[7]=jmpflag;

Nx=nx[i];Ny=ny[i];Nxp=nxp;Nyp=nyp;Nxm=nxm;Nym=nym;
C1=c1;C2=c2;C3=c3;C4=c4;C5=c5;C7=c7;
/*if(nx[i]==0 && ny[i]==0){
printf(" %d %d\n",x[1],y[1]);
```

```
putpixel_2(1,0,0);printf(" %d %d\n",X_,Y_);
nv(1,c3,3);printf(" %d %d\n",x_[1],y_[1]);
}*/



if((c1==ca)||(c2==ca)||(c3==ca)||(c4==ca)||(c5==ca)||(c7==ca)){
s.xx=nx[i];s.yy=ny[i];s.xx_=nx_[i];s.yy_=ny_[i];fwrite_mem(i);

std_x=nx[i];std_y=ny[i];            /* S */
last_x=nx_[i];last_y=ny_[i];
nx_[i]=nx[i];ny_[i]=ny[i];          /* new b */



#if ALGO==0
if(algo==0){
val=-5;
if((/*c1!=ca*/nv(0,c1,1)==1)&&(c5==ca)){  /* CW (no phase) */
  if((angle=getangle(nxp,nyp))>val) {val=angle;nx[i]=x[5];ny[i]=y[5];jp=jmp[5];}}
if((/*c5!=ca*/nv(0,c5,5)==1)&&(c2==ca)){
  if((angle=getangle(std_x,nyp))>val) {val=angle;nx[i]=x[2];ny[i]=y[2];jp=jmp[2];}}
if((/*c2!=ca*/nv(0,c2,2)==1)&&(c3==ca)){
  if((angle=getangle(nxm,std_y))>val) {val=angle;nx[i]=x[3];ny[i]=y[3];jp=jmp[3];}}
if((/*c3!=ca*/nv(0,c3,3)==1)&&(c7==ca)){
  if((angle=getangle(nxm,nym))>val) {val=angle;nx[i]=x[7];ny[i]=y[7];jp=jmp[7];}}
if((/*c7!=ca*/nv(0,c7,7)==1)&&(c4==ca)){
  if((angle=getangle(std_x,nym))>val) {val=angle;nx[i]=x[4];ny[i]=y[4];jp=jmp[4];}}
if((/*c4!=ca*/nv(0,c4,4)==1)&&(c1==ca)){
  if((angle=getangle(nxp,std_y))>val) {val=angle;nx[i]=x[1];ny[i]=y[1];jp=jmp[1];}}
if(val==-5) /*beep(10000)*/;
}
else{
val=5;
if((/*c1!=ca*/nv(1,c1,1)==1)&&(c4==ca)){  /* CCW (no phase) */
  if((angle=getangle(std_x,nym))<val) {val=angle;nx[i]=x[4];ny[i]=y[4];jp=jmp[4];}}
if((/*c4!=ca*/nv(1,c4,4)==1)&&(c7==ca)){
  if((angle=getangle(nxm,nym))<val) {val=angle;nx[i]=x[7];ny[i]=y[7];jp=jmp[7];}}
if((/*c7!=ca*/nv(1,c7,7)==1)&&(c3==ca)){
  if((angle=getangle(nxm,std_y))<val) {val=angle;nx[i]=x[3];ny[i]=y[3];jp=jmp[3];}}
if((/*c3!=ca*/nv(1,c3,3)==1)&&(c2==ca)){
  if((angle=getangle(std_x,nyp))<val) {val=angle;nx[i]=x[2];ny[i]=y[2];jp=jmp[2];}}
if((/*c2!=ca*/nv(1,c2,2)==1)&&(c5==ca)){
  if((angle=getangle(nxp,nyp))<val) {val=angle;nx[i]=x[5];ny[i]=y[5];jp=jmp[5];}}
if((/*c5!=ca*/nv(1,c5,5)==1)&&(c1==ca)){
  if((angle=getangle(nxp,std_y))<val) {val=angle;nx[i]=x[1];ny[i]=y[1];jp=jmp[1];}}
```

```
if(val==5) /*beep(10000)*/;
}

if(jp){
tmp=nx_[i];nx_[i]=ny_[i];ny_[i]=tmp;
}
#else
if(algo==0){
if(i==0){
/* al0:0-1-2-3-4-5 */
     if(/*c4!=ca*/nv(0,c4,4)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];al0=0;}  /* CW */
else if(/*c1!=ca*/nv(0,c1,1)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];al0=1;}
else if(/*c5!=ca*/nv(0,c5,5)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];al0=2;}
else if(/*c2!=ca*/nv(0,c2,2)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];al0=3;}
else if(/*c3!=ca*/nv(0,c3,3)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];al0=4;}
else if(/*c7!=ca*/nv(0,c7,7)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];al0=5;}
}
else{
if(flag_lr==0) ali=fw[0][al0][i];
else           ali=fw[1][al0][i];

if(ali==0){
     if(/*c4!=ca*/nv(0,c4,4)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CW */
else if(/*c1!=ca*/nv(0,c1,1)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(0,c5,5)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(0,c2,2)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(0,c3,3)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(0,c7,7)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
}
else if(ali==1){
     if(/*c1!=ca*/nv(0,c1,1)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(0,c5,5)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(0,c2,2)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(0,c3,3)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(0,c7,7)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(0,c4,4)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CW */
}
else if(ali==2){
     if(/*c5!=ca*/nv(0,c5,5)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(0,c2,2)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(0,c3,3)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(0,c7,7)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(0,c4,4)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CW */
else if(/*c1!=ca*/nv(0,c1,1)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
}
else if(ali==3){
```

```
     if(/*c2!=ca*/nv(0,c2,2)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(0,c3,3)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(0,c7,7)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(0,c4,4)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CW */
else if(/*c1!=ca*/nv(0,c1,1)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(0,c5,5)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
}
else if(ali==4){
     if(/*c3!=ca*/nv(0,c3,3)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(0,c7,7)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(0,c4,4)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CW */
else if(/*c1!=ca*/nv(0,c1,1)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(0,c5,5)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(0,c2,2)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
}
else if(ali==5){
     if(/*c7!=ca*/nv(0,c7,7)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(0,c4,4)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CW */
else if(/*c1!=ca*/nv(0,c1,1)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(0,c5,5)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(0,c2,2)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(0,c3,3)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
}

}
}
else{
if(i==0){
/* al0:0-5-4-3-2-1 */
     if(/*c5!=ca*/nv(1,c5,5)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];al0=0;}  /* CCW */
else if(/*c1!=ca*/nv(1,c1,1)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];al0=5;}
else if(/*c4!=ca*/nv(1,c4,4)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];al0=4;}
else if(/*c7!=ca*/nv(1,c7,7)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];al0=3;}
else if(/*c3!=ca*/nv(1,c3,3)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];al0=2;}
else if(/*c2!=ca*/nv(1,c2,2)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];al0=1;}
}
else{
if(flag_lr==0) ali=fw[0][al0][i];
else           ali=fw[1][al0][i];

if(ali==0){
     if(/*c5!=ca*/nv(1,c5,5)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CCW */
else if(/*c1!=ca*/nv(1,c1,1)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(1,c4,4)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(1,c7,7)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(1,c3,3)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
```

```
else if(/*c2!=ca*/nv(1,c2,2)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
}
else if(ali==5){
     if(/*c1!=ca*/nv(1,c1,1)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(1,c4,4)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(1,c7,7)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(1,c3,3)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(1,c2,2)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(1,c5,5)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CCW */
}
else if(ali==4){
     if(/*c4!=ca*/nv(1,c4,4)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(1,c7,7)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(1,c3,3)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(1,c2,2)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(1,c5,5)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CCW */
else if(/*c1!=ca*/nv(1,c1,1)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
}
else if(ali==3){
     if(/*c7!=ca*/nv(1,c7,7)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(1,c3,3)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(1,c2,2)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(1,c5,5)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CCW */
else if(/*c1!=ca*/nv(1,c1,1)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(1,c4,4)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
}
else if(ali==2){
     if(/*c3!=ca*/nv(1,c3,3)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
else if(/*c2!=ca*/nv(1,c2,2)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(1,c5,5)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CCW */
else if(/*c1!=ca*/nv(1,c1,1)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(1,c4,4)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(1,c7,7)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
}
else if(ali==1){
     if(/*c2!=ca*/nv(1,c2,2)==1 && c5==ca) {nx[i]=x[5];ny[i]=y[5];}
else if(/*c5!=ca*/nv(1,c5,5)==1 && c1==ca) {nx[i]=x[1];ny[i]=y[1];}  /* CCW */
else if(/*c1!=ca*/nv(1,c1,1)==1 && c4==ca) {nx[i]=x[4];ny[i]=y[4];}
else if(/*c4!=ca*/nv(1,c4,4)==1 && c7==ca) {nx[i]=x[7];ny[i]=y[7];}
else if(/*c7!=ca*/nv(1,c7,7)==1 && c3==ca) {nx[i]=x[3];ny[i]=y[3];}
else if(/*c3!=ca*/nv(1,c3,3)==1 && c2==ca) {nx[i]=x[2];ny[i]=y[2];}
}

}
}
#endif
```

```
if(1) putpixel_(nx[i],ny[i],acolor[i]);
flag_pp[i]=1;
}/**if(c1,c2,c3,c4)**/
else{
if(ftell_mem(i)==0) {flag_[i]=0;cp--;if(cp==0) break;}
fread_mem(i);
nx[i]=s.xx;ny[i]=s.yy;nx_[i]=s.xx_;ny_[i]=s.yy_;
flag_pp[i]=0;
}/**else(c1,c2,c3,c4)**/
}/**if(flag_[i])**/

i++;if(i==CPMAX) break;
}/**while(1)**/

if(0){
i=0;
while(1){
ig=i;
if(flag_[i]==1 && flag_pp[i]==1) putpixel_(nx[i],ny[i],acolor[i]);

i++;if(i==CPMAX) break;
}/**while(1)**/
}

if(flag_us==0 && flag_[0]==1 && flag_pp[0]==1 && check_len(nx_old,ny_old)>1){
printf(" ?\n");
i=0;putpixel(nx_[i],ny_[i],12);
i=1;putpixel(nx_[i],ny_[i],12);
i=2;putpixel(nx_[i],ny_[i],12);
use_subroop();flag_us=1;
}
}/**while(cp)**/

return 0;
}/** cag_r **/
```