Cellular Automaton Graphics(10)

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. Infinite plane(square pixel)

Figure 1 shows a domain in which square pixel is used for tiling of infinite plane.

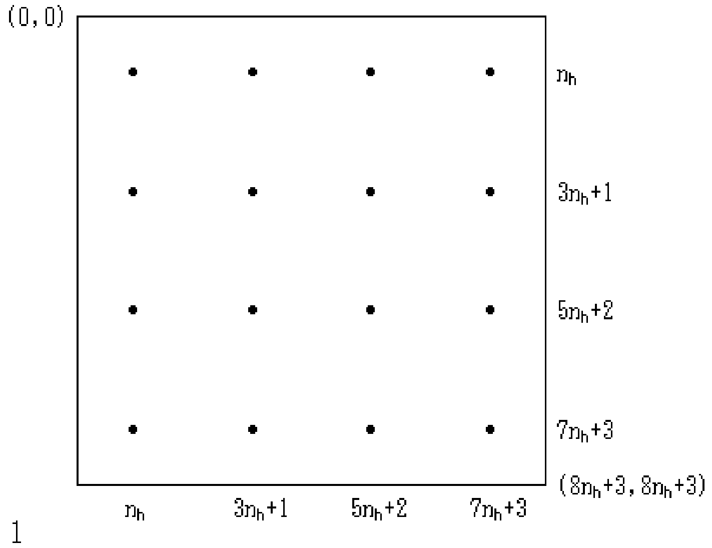

Figure 1

The points in the figure are seed point. Assuming n_s to be (number of square pixels between two seed points)/2, the coordinates of the seed points are as follows:

- $(n_\mathrm{s},\ n_\mathrm{s})$, $(3n_\mathrm{s}+1,\ n_\mathrm{s})$, $(5n_\mathrm{s}+2,\ n_\mathrm{s})$, $(7n_\mathrm{s}+3,\ n_\mathrm{s})$
- $(n_\mathrm{s},\ 3n_\mathrm{s}+1)$, $(3n_\mathrm{s}+1,\ 3n_\mathrm{s}+1)$, $(5n_\mathrm{s}+2,\ 3n_\mathrm{s}+1)$, $(7n_\mathrm{s}+3,\ 3n_\mathrm{s}+1)$
- $(n_\mathrm{s},\ 5n_\mathrm{s}+2)$, $(3n_\mathrm{s}+1,\ 5n_\mathrm{s}+2)$, $(5n_\mathrm{s}+2,\ 5n_\mathrm{s}+2)$, $(7n_\mathrm{s}+3,\ 5n_\mathrm{s}+2)$
- $(n_\mathrm{s},\ 7n_\mathrm{s}+3)$, $(3n_\mathrm{s}+1,\ 7n_\mathrm{s}+3)$, $(5n_\mathrm{s}+2,\ 7n_\mathrm{s}+3)$, $(7n_\mathrm{s}+3,\ 7n_\mathrm{s}+3)$
- $n_\mathrm{s} > 0$

In a program, Lsqr is used instead of n_s.

A painting point next to seed point is decided choosing the amount of change of coordinates, for example, as follows:

- $(n_\mathrm{s},\ n_\mathrm{s})$, $(3n_\mathrm{s}+1,\ n_\mathrm{s})$, $(5n_\mathrm{s}+2,\ n_\mathrm{s})$, $(7n_\mathrm{s}+3,\ n_\mathrm{s})$：$\Delta x = 1$
- $(n_\mathrm{s},\ 3n_\mathrm{s}+1)$, $(3n_\mathrm{s}+1,\ 3n_\mathrm{s}+1)$, $(5n_\mathrm{s}+2,\ 3n_\mathrm{s}+1)$, $(7n_\mathrm{s}+3,\ 3n_\mathrm{s}+1)$：$\Delta x = 1$
- $(n_\mathrm{s},\ 5n_\mathrm{s}+2)$, $(3n_\mathrm{s}+1,\ 5n_\mathrm{s}+2)$, $(5n_\mathrm{s}+2,\ 5n_\mathrm{s}+2)$, $(7n_\mathrm{s}+3,\ 5n_\mathrm{s}+2)$：$\Delta x = 1$
- $(n_\mathrm{s},\ 7n_\mathrm{s}+3)$, $(3n_\mathrm{s}+1,\ 7n_\mathrm{s}+3)$, $(5n_\mathrm{s}+2,\ 7n_\mathrm{s}+3)$, $(7n_\mathrm{s}+3,\ 7n_\mathrm{s}+3)$：$\Delta x = 1$
- $\Delta y = 0$

- $(n_\mathrm{s},\ n_\mathrm{s})$, $(3n_\mathrm{s}+1,\ n_\mathrm{s})$, $(5n_\mathrm{s}+2,\ n_\mathrm{s})$, $(7n_\mathrm{s}+3,\ n_\mathrm{s})$：$\Delta x = 1$
- $(n_\mathrm{s},\ 3n_\mathrm{s}+1)$, $(3n_\mathrm{s}+1,\ 3n_\mathrm{s}+1)$, $(5n_\mathrm{s}+2,\ 3n_\mathrm{s}+1)$, $(7n_\mathrm{s}+3,\ 3n_\mathrm{s}+1)$：$\Delta x = -1$

- $(n_s,\ 5n_s+2),\ (3n_s+1,\ 5n_s+2),\ (5n_s+2,\ 5n_s+2),\ (7n_s+3,\ 5n_s+2) : \Delta x = 1$
- $(n_s,\ 7n_s+3),\ (3n_s+1,\ 7n_s+3),\ (5n_s+2,\ 7n_s+3),\ (7n_s+3,\ 7n_s+3) : \Delta x = -1$
- $\Delta y = 0$

or

- $(n_s,\ n_s) : \Delta x = 1,\ (3n_s+1,\ n_s) : \Delta x = -1,\ (5n_s+2,\ n_s) : \Delta x = 1,$
  $(7n_s+3,\ n_s) : \Delta x = -1$
- $(n_s,\ 3n_s+1) : \Delta x = -1,\ (3n_s+1,\ 3n_s+1) : \Delta x = 1,\ (5n_s+2,\ 3n_s+1) : \Delta x = -1,$
  $(7n_s+3,\ 3n_s+1) : \Delta x = 1$
- $(n_s,\ 5n_s+2) : \Delta x = 1,\ (3n_s+1,\ 5n_s+2) : \Delta x = -1,\ (5n_s+2,\ 5n_s+2) : \Delta x = 1,$
  $(7n_s+3,\ 5n_s+2) : \Delta x = -1$
- $(n_s,\ 7n_s+3) : \Delta x = -1,\ (3n_s+1,\ 7n_s+3) : \Delta x = 1,\ (5n_s+2,\ 7n_s+3) : \Delta x = -1,$
  $(7n_s+3,\ 7n_s+3) : \Delta x = 1$
- $\Delta y = 0$

They are called parallelism, antiparallelism and oblique antiparallelism respectively.

We use CW in Figure 5 and CCW in Figure 6 in the 6th as painting algorithm. If a painting point projects out of the domain, it jumps under periodic boundary condition.

Figure 2 is a sample of painting.

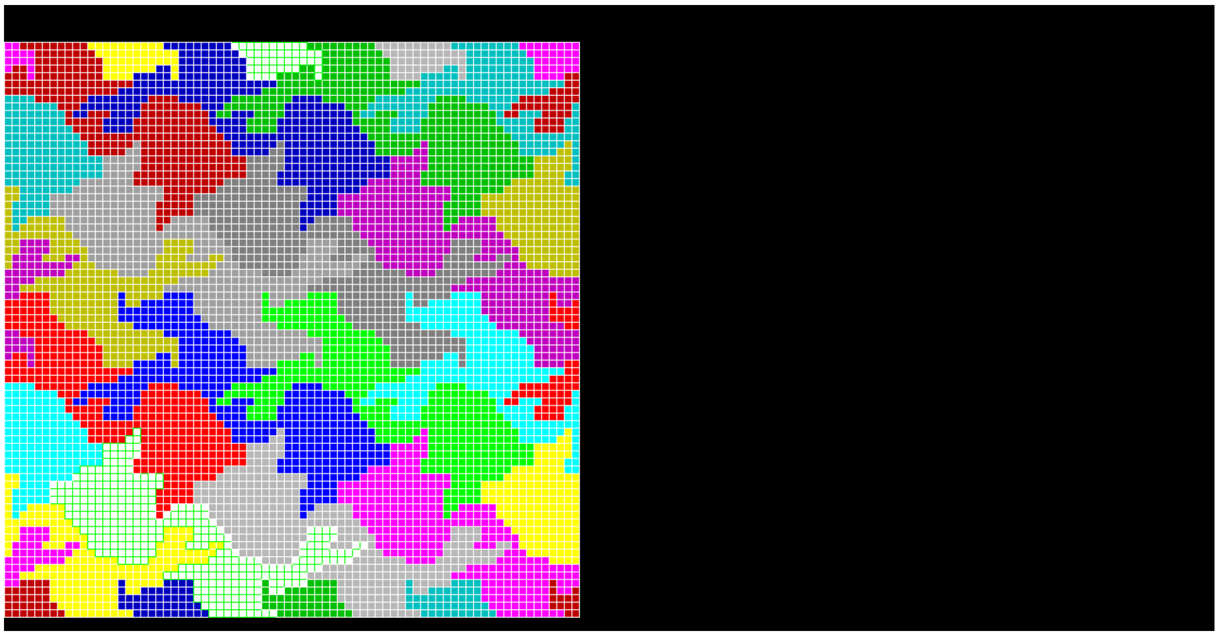
Figure 2

2. Infinite plane(hexagonal pixel)

Figure 3 shows a domain D_1 in which hexagonal pixel is used for tiling of infinite plane. The black pixel in the figure is Dead Pixel of D_1.

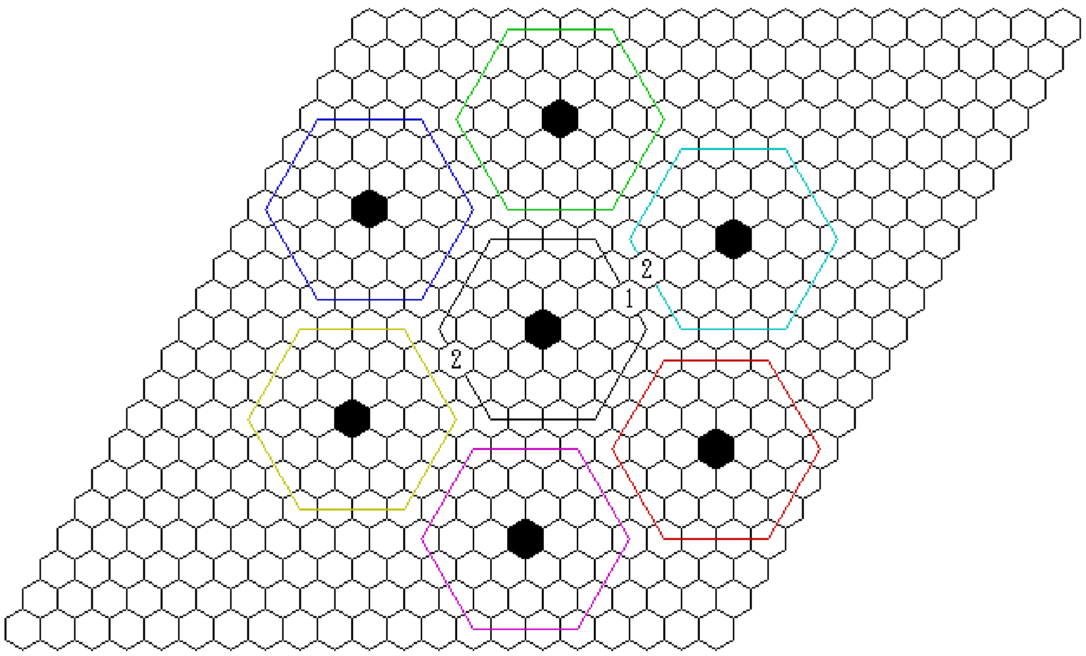

Figure 3

Fogure 4 shows a domain D_7 which is a combination of 7 D_1s.

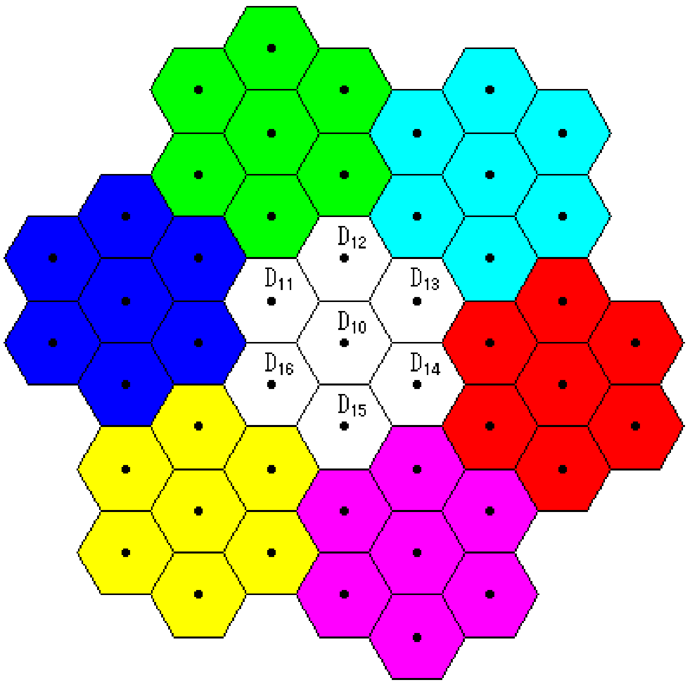

Figure 4

The point in the figure is Dead Pixel of D_1. Assuming n_h to be number of pixels of a side of D_1, the coordinates of Dead Pixels are as follows:

- D_{10}：$(3(n_h-1)+2 \equiv X_0,\ 3(n_h-1)+2 \equiv Y_0)$ or $(3(n_h-1)+1 \equiv X_0,\ 3(n_h-1)+1 \equiv Y_0)$
- D_{11}：$(X_0-2(n_h-1)-1,\ Y_0-(n_h-1)-1)$
- D_{12}：$(X_0-(n_h-1),\ Y_0-2(n_h-1)-1)$
- D_{13}：$(X_0+n_h,\ Y_0-(n_h-1))$
- D_{14}：$(X_0+2(n_h-1)+1,\ Y_0+n_h)$
- D_{15}：$(X_0+(n_h-1),\ Y_0+2(n_h-1)+1)$
- D_{16}：$(X_0-n_h,\ Y_0+(n_h-1))$

Using the symbols in Figure 4 in the 4th, the coordinates of vertexes of D_{10} are as follows:

- v0 : $(X_0 - (n_\mathrm{h} - 1),\ Y_0 - (n_\mathrm{h} - 1))$
- v1 : $(X_0,\ Y_0 - (n_\mathrm{h} - 1))$
- v2 : $(X_0 + (n_\mathrm{h} - 1),\ Y_0)$
- v3 : $(X_0 + (n_\mathrm{h} - 1),\ Y_0 + (n_\mathrm{h} - 1))$
- v4 : $(X_0,\ Y_0 + (n_\mathrm{h} - 1))$
- v5 : $(X_0 - (n_\mathrm{h} - 1),\ Y_0)$

In a program, Lhex is used instead of n_h.
Including Dead Pixel, the number of pixels in D_1, D_7 is

- D_1 : $3{n_\mathrm{h}}^2 - 3n_\mathrm{h} + 1 \equiv N_1$
- D_7 : $7N_1$

Seed points are arranged by 6 pieces around a Dead Pixel like Figure 5 for example. The pixel with a prime is a painting point next to seed point. Seed points and painting points next to seed point are decided under 6-fold symmetry.

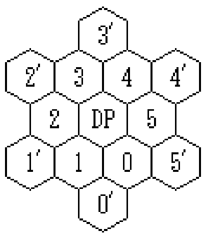

Figure 5

We use CW in Figure 11 and CCW in Figure 12 in the 1st as painting algorithm. If a painting point projects out of the domain, it jumps under periodic boundary condition. On D_1, in Figure 3, for example, if a painting point moves from 1 to 2, it jumps from the 2 to the other 2. On D_7, in Figure 4, get a D_1 of jump destination referring the arrangement of outer D_1s in Figure 3.

Figure 6 is a sample of painting.

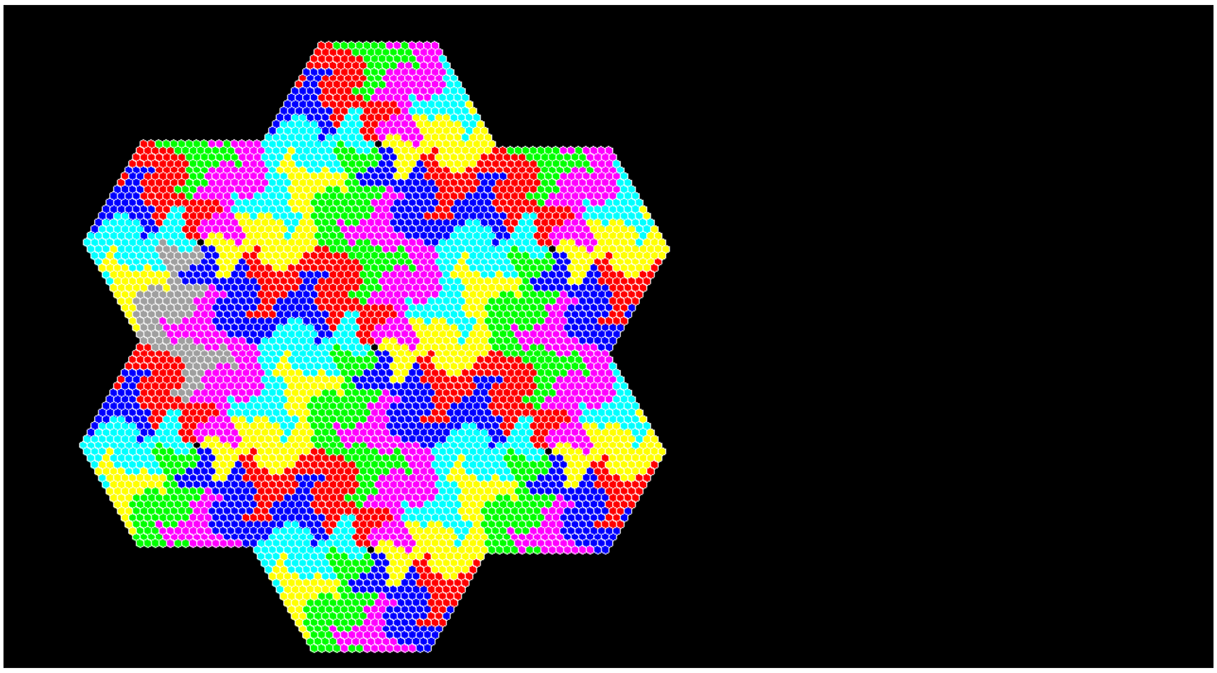

Figure 6

3. Tiling by handiwork

We consider making of a regular figure by not switching of CW, CCW but handiwork. In cellular automaton graphics, we deal with the following domain.

- infinite plane(square pixel)
- infinite plane(hexagonal pixel)
- polyhedron
- 3-dimensional space
- multidimensional space

We had used switching of CW, CCW from the 1st through the 9th on polyhedron.

4. Infinite plane(square pixel)

The easiest method when we make figures regularly by handiwork in a domain like Figure 1 is parallel shift method. We make a figure in Figure 7 by handiwork namely with painting through mouse. The figure is called seed figure.

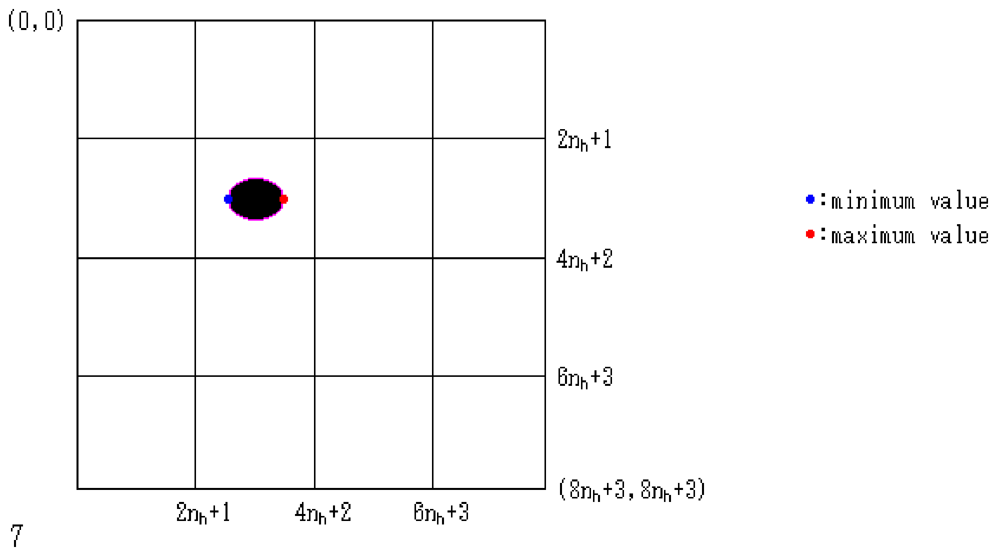

Figure 7

A pixel which constitutes the seed figure is copied vertically and horizontally. We assume the maximum value and the minimum value of the seed figure in the direction of x to be $x_{\min}$ and $x_{\max}$ respectively. In the figure, $x_{\min}$ is in the following range.

- $2n_s + 1 \leq x_{\min} < 4n_s + 2$

Therefore, the number of times of copies of the seed figure is 1 and 2 in left and right respectively. If $x_{\max}$ is in the following range, the part which projects is moved to left end at the second time copy.

- $4n_s + 2 \leq x_{\max}$

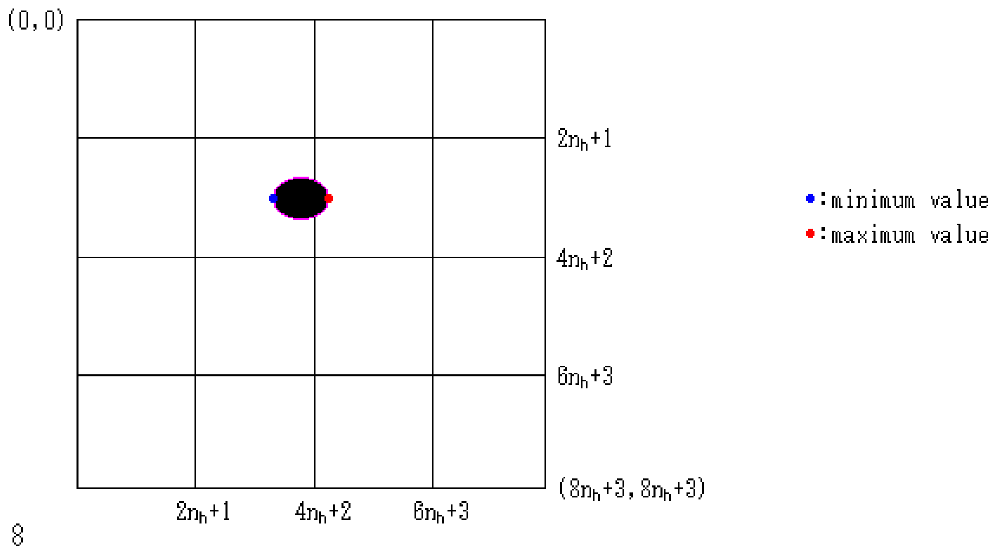

Figure 8

The same copy is done in the direction of y.

Parallel shift method is valid if only orthogonal coordinate system is used. For example, in 3-dimensional space, the above copy is done in the direction of x, y, z.

5. Infinite plane(hexagonal pixel)

Painting number method and painting detection method is used in the domain like Figure 3. In painting number method, before doing of tiling by handiwork, while making a figure like Figure 6 with only CW or only CCW, we memorize the coordinates of pixels which are painted into painting number array every painting number. Painting number is as follows:

- seed point(6×7 pieces) : 0
- painting point next to seed point(6×7 pieces) : 1

  . . . .
- last painting point(6×7 pieces) : number of pixels which were painted on one painting point-1

We make a seed figure in Figure 9 with painting through mouse.

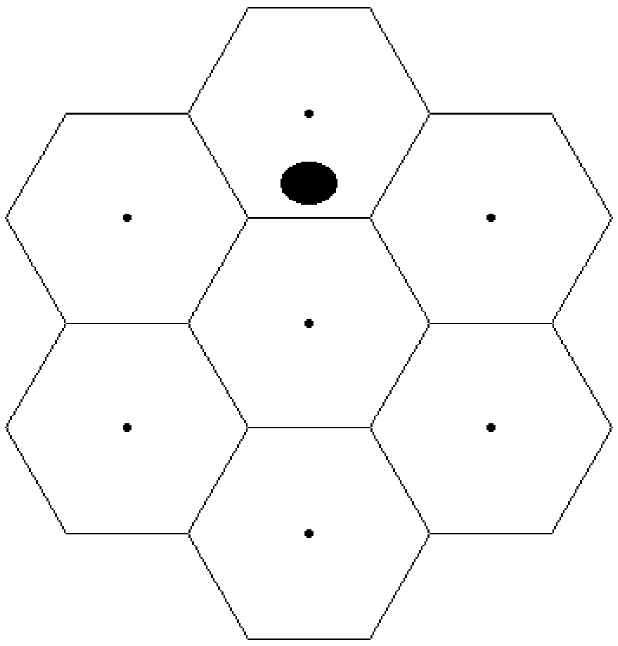
Figure 9

Searching for coordinates of a pixel which constitutes the seed figure from painting number array, first, we paint the pixel and next, reading coordinates of elements($6 \times 7 - 1$ pieces) of painting number array which are the same as the pixel in painting number, we paint the pixels. We do the work on all the pixels which constitute the seed figure.

In painting detection method, while making a figure like Figure 6 with only CW or only CCW, if a painting point detects coordinates of a pixel which constitutes the seed figure, we memorize the coordinates of all the then painting points namely members into an array and after the making of a figure, we make domains for painting reading the coordinates of elements of the array.

In Figure 10, the seed figure is a border like closed curve. In this case, we make domains for painting copying the border with painting detection method.

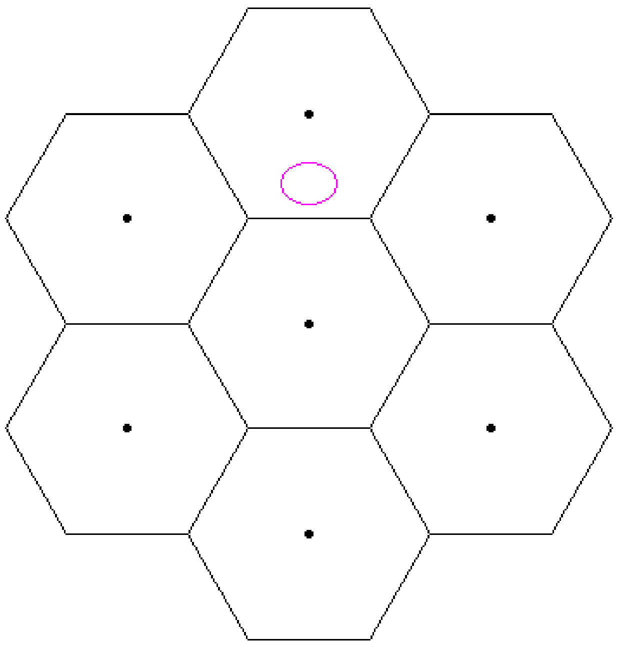

Figure 10

After the making of domains for painting, copying seed points into the domains with painting number method, we do paintings.

6. Painting algorithm(square pixel)

In painting of square pixel, we use two types of algorithm which are different in number of connections. Figure 11 is a 8-connected border made by mouse action. This border must be closed.

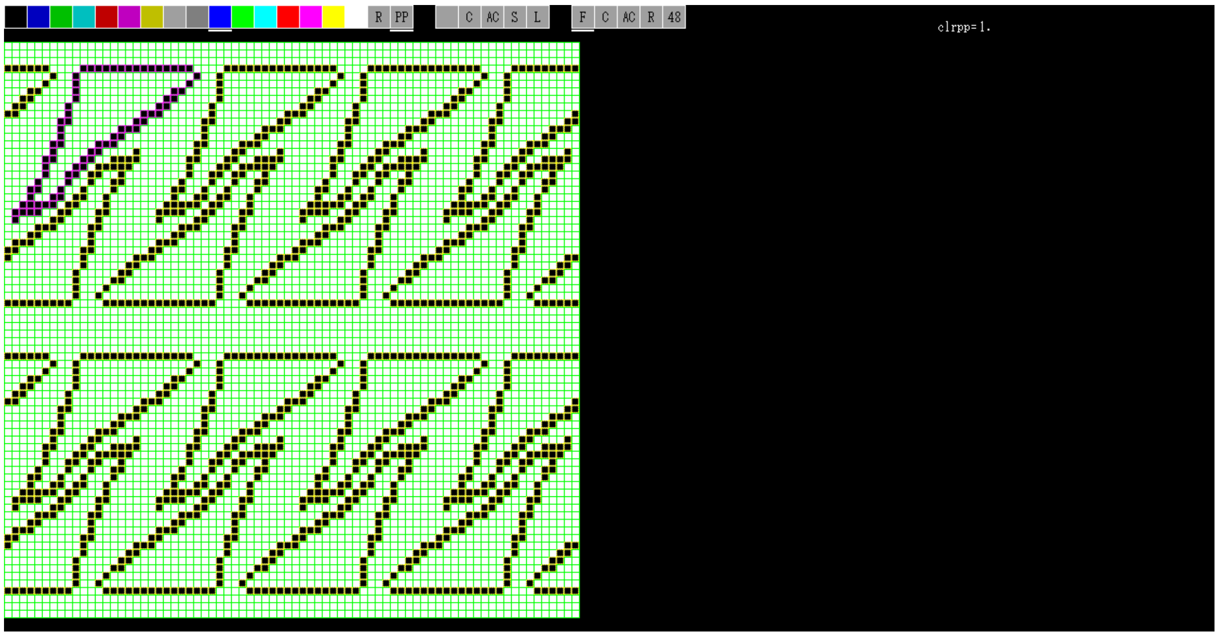

Figure 11

Copying the border with painting detection method, if we do painting of the inside of the border, C+S in Figure 12, C+0 in Figure 13 and 0+S in Figure 14 are got. C is border and S is the inside. Figure 15 is used in painting algorithm. If we detect the pixel of the border when painting of S is done and we paint the pixel, C is got.

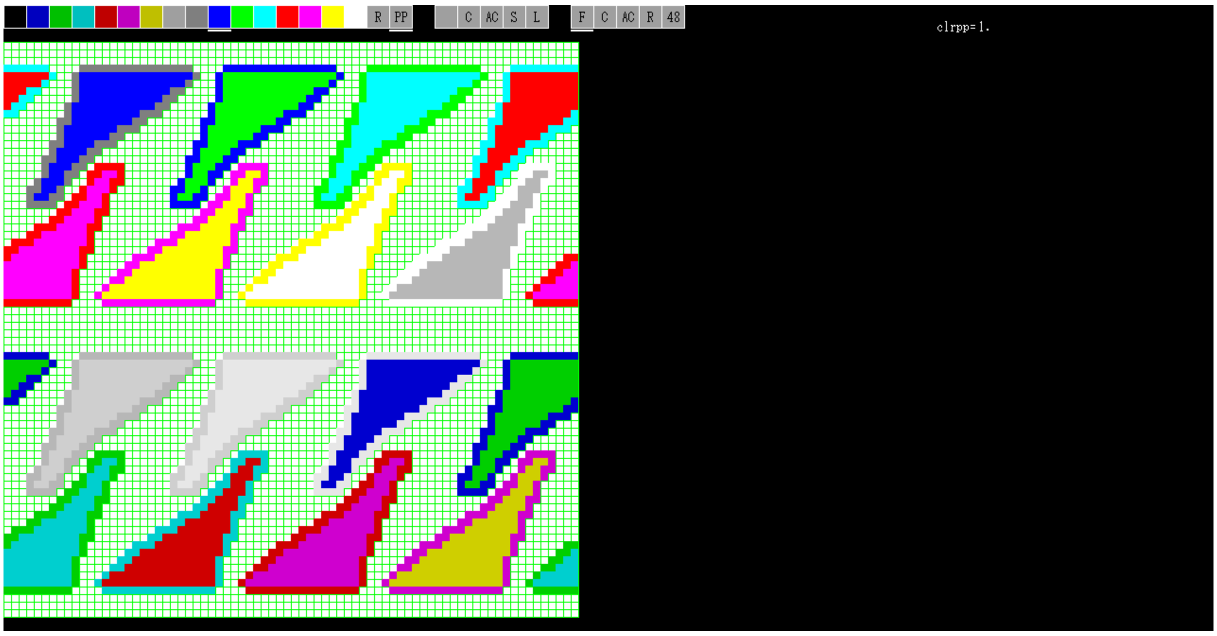

Figure 12

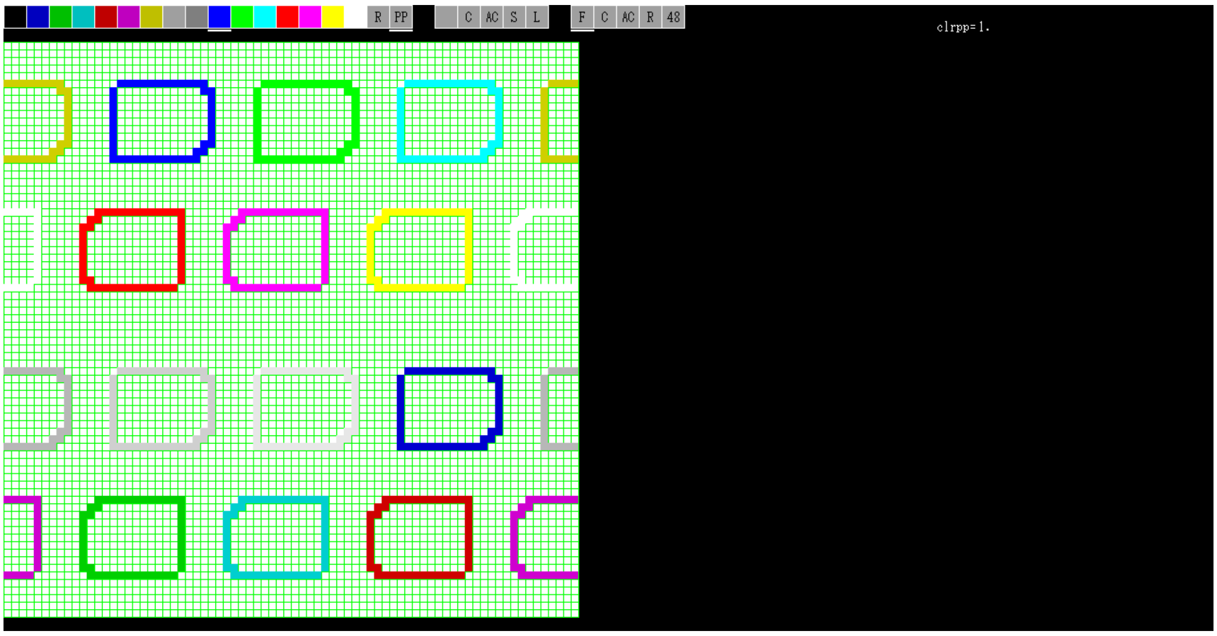

Figure 13

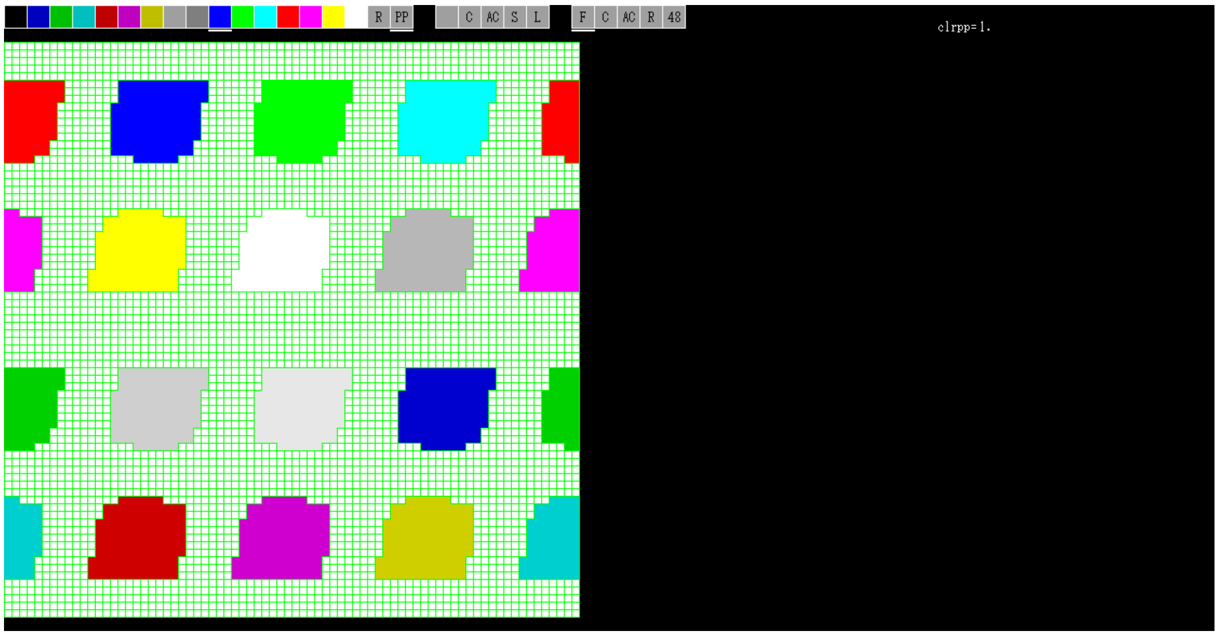

Figure 14

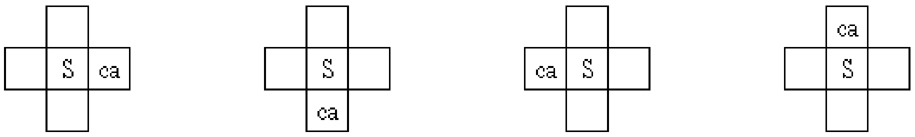

Figure 15

We assume S of C+S in Figure 12 to be seed figure. Copying the seed figure with painting detection method, if we do painting, S′ in Figure 16 is got. Figure 15 is used in painting algorithm.

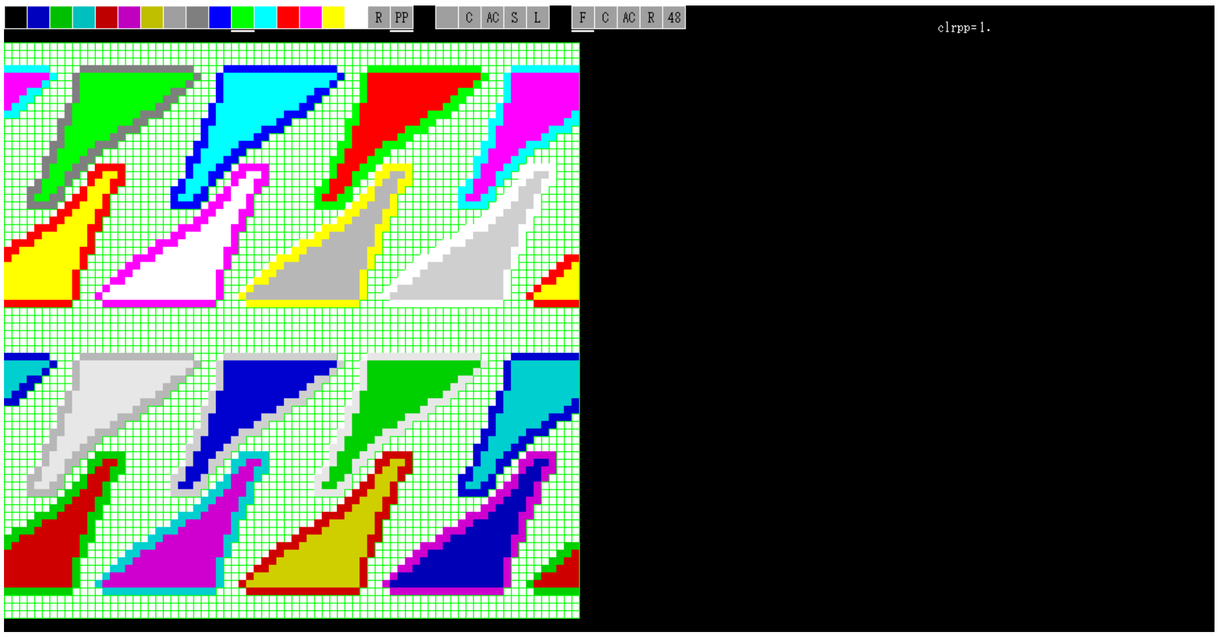

Figure 16

Figure 17 is a 8-connected figure made by mouse action.

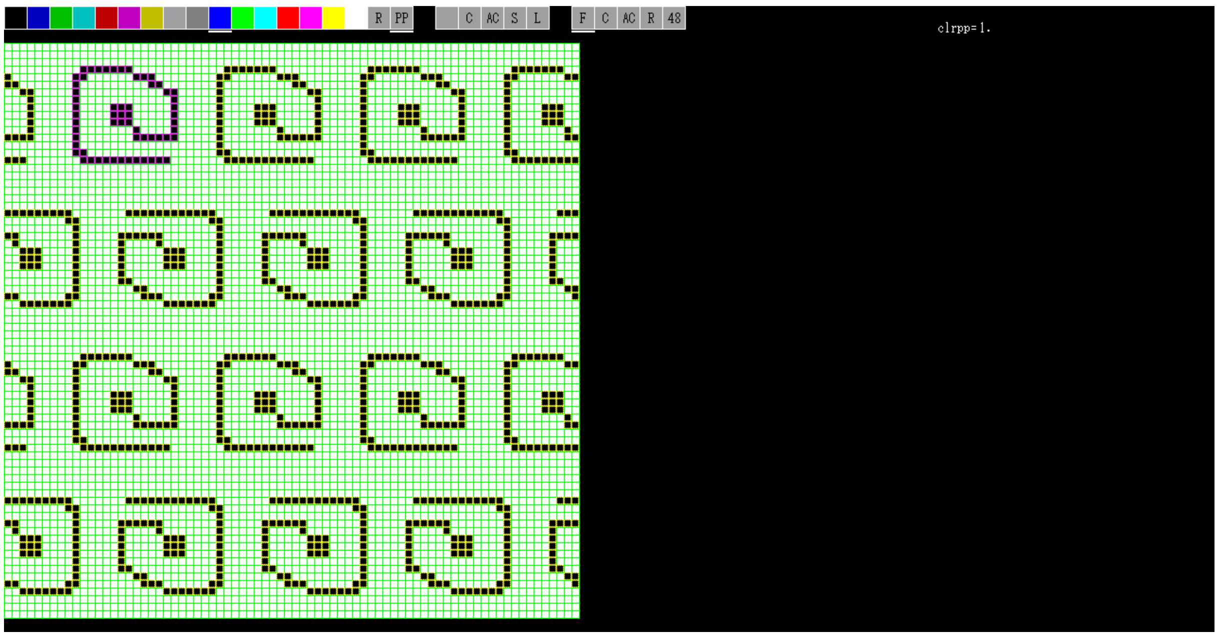

Figure 17

Copying the figure with painting detection method, if we do painting, C′ in Figure 18 is got. Figure 19 is used in painting algorithm.

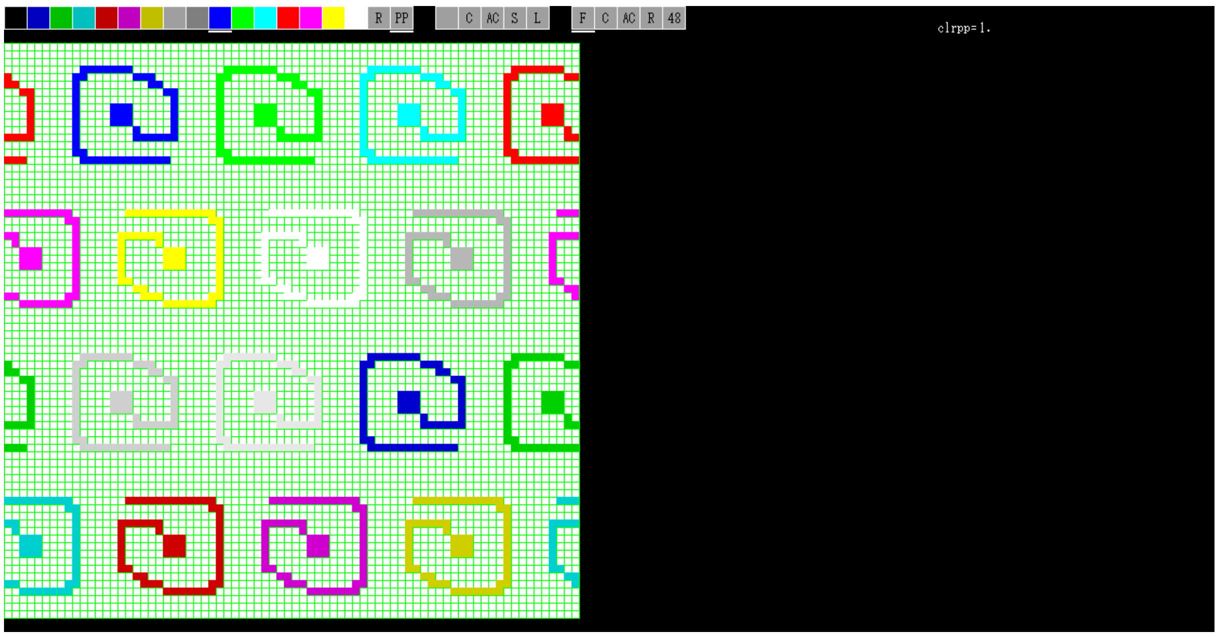

Figure 18

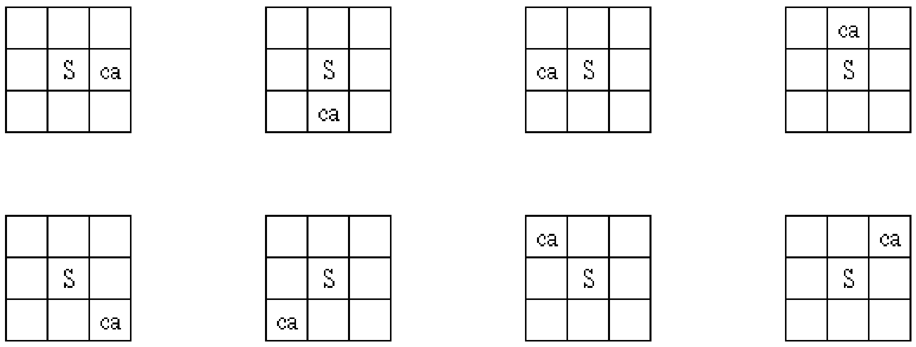

Figure 19

7. Painting algorithm(hexagonal pixel)

In painting of hexagonal pixel, we use the algorithm shown in Figure 20.

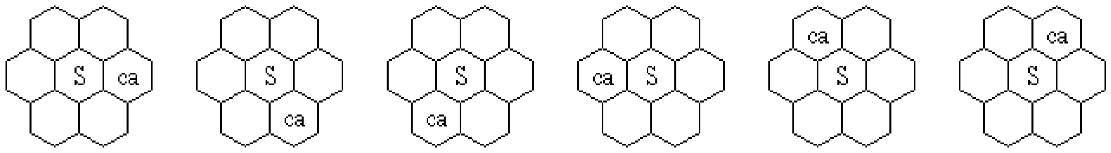

Figure 20

Figure 21 is a 6-connected border made by mouse action. This border must be closed.

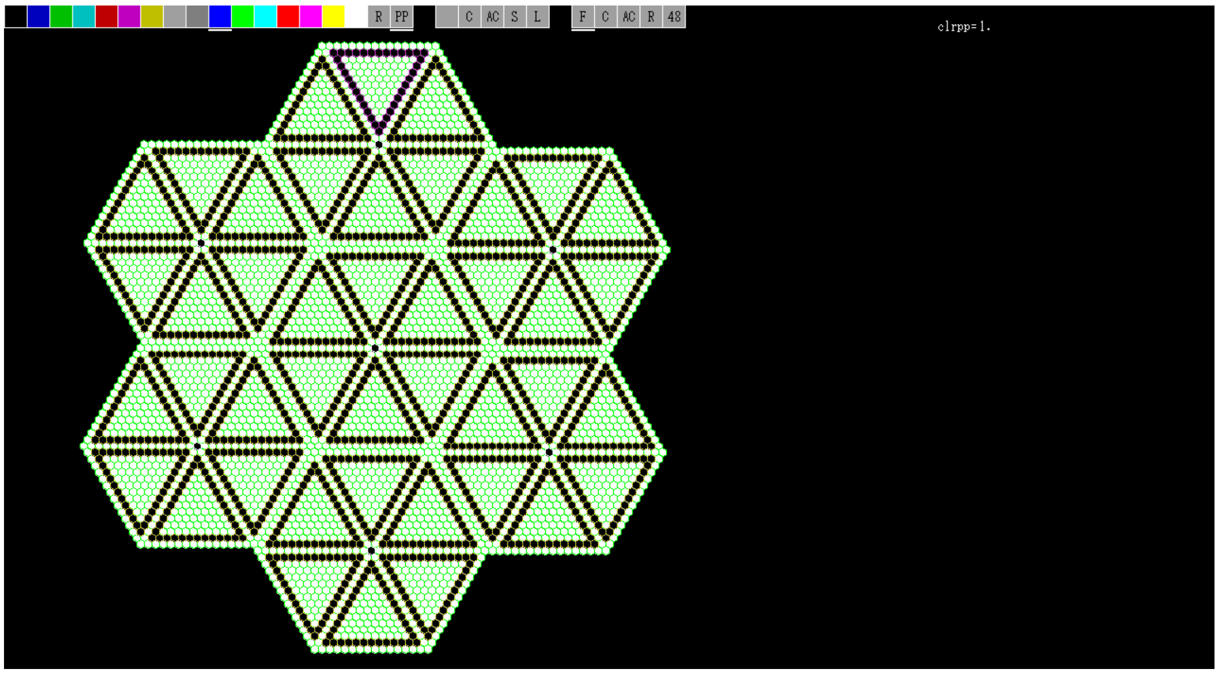

Figure 21

Copying the border with painting detection method, if we do painting of the inside of the border, C+S in Figure 22, C+0 in Figure 23 and 0+S in Figure 24 are got. C is border and S is the inside. If we detect the pixel of the border when painting of S is done and we paint the pixel, C is got.

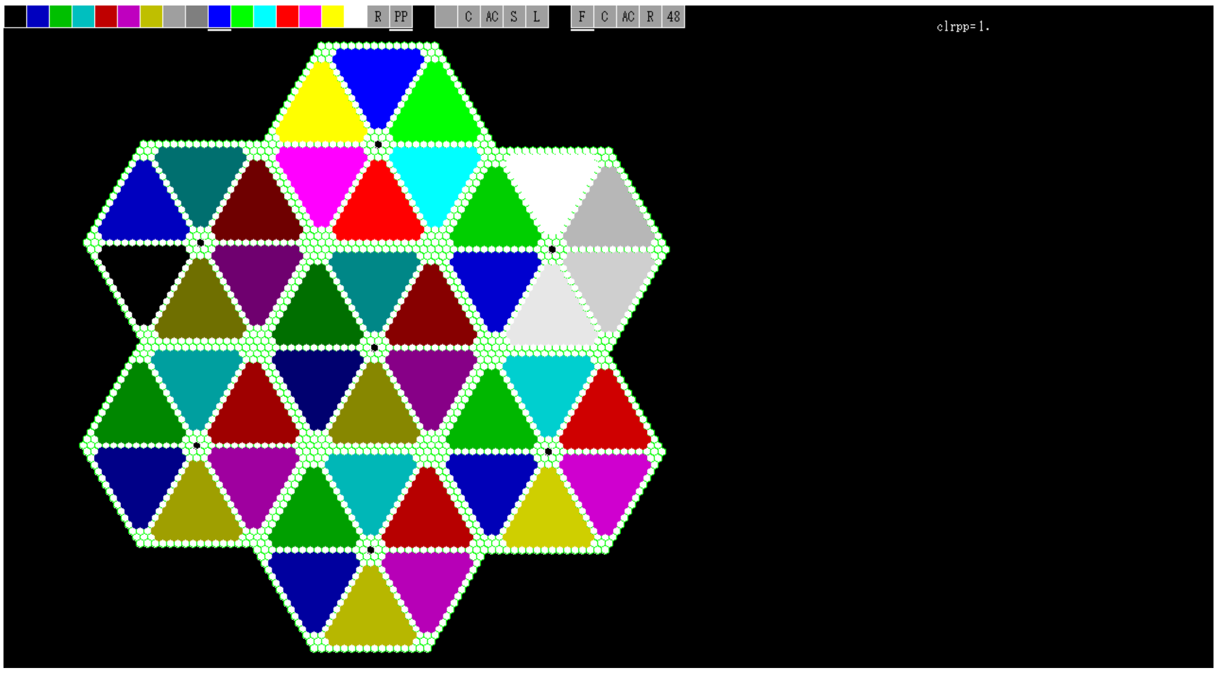

Figure 22

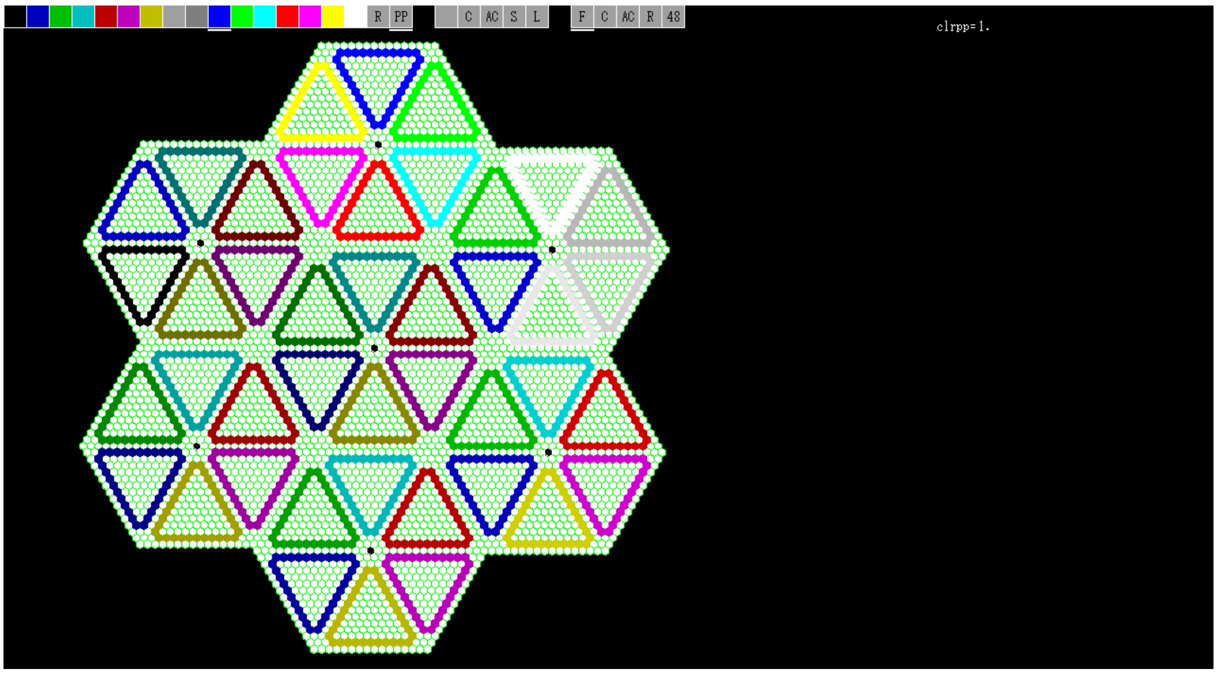

Figure 23

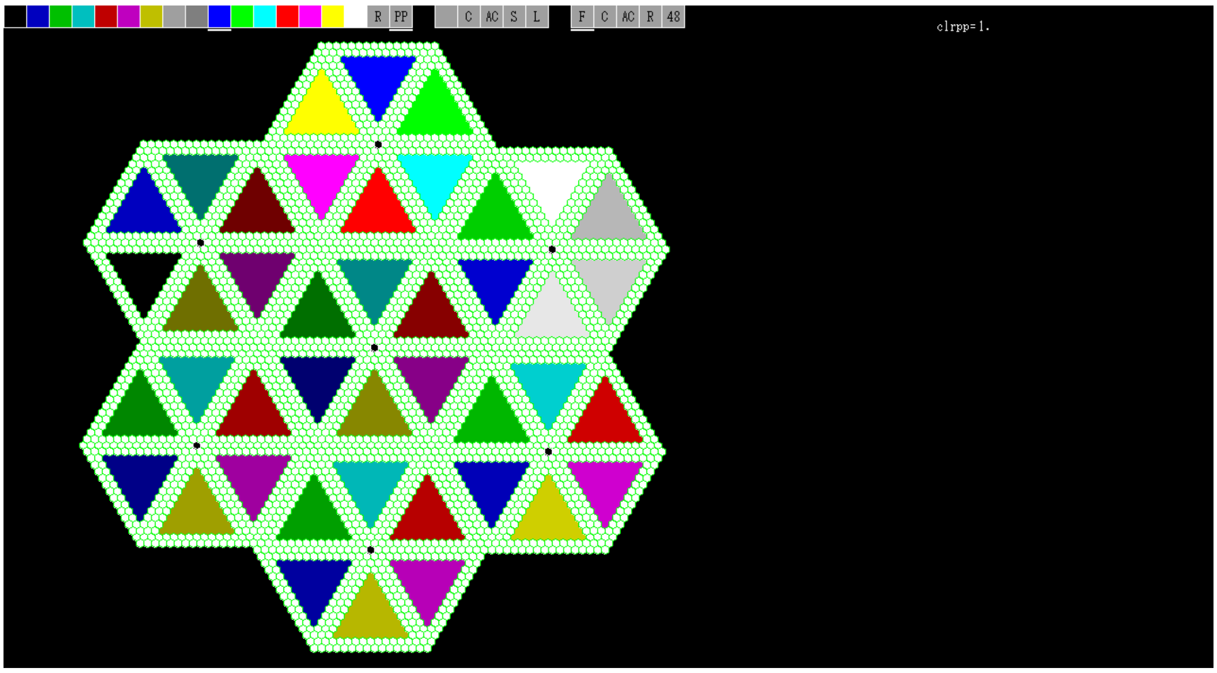

Figure 24

We assume S of C+S in Figure 22 to be seed figure. Copying the seed figure with painting detection method, if we do painting, S′ in Figure 25 is got.

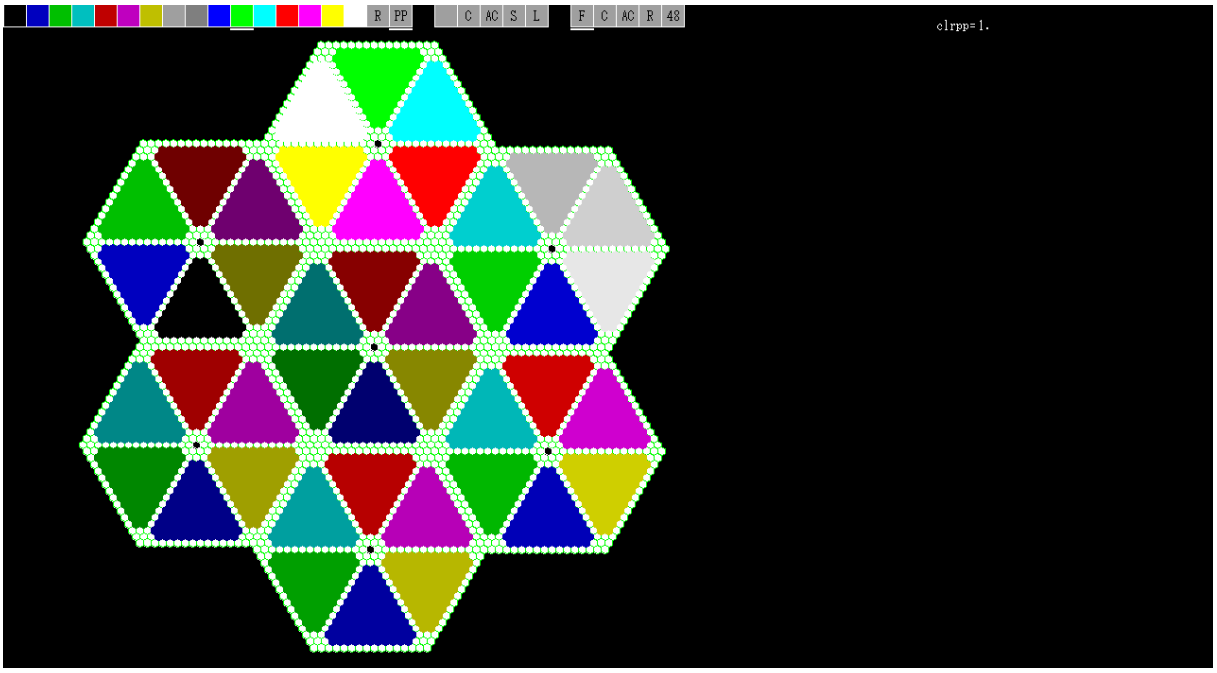

Figure 25

Figure 26 is a 6-connected figure made by mouse action.

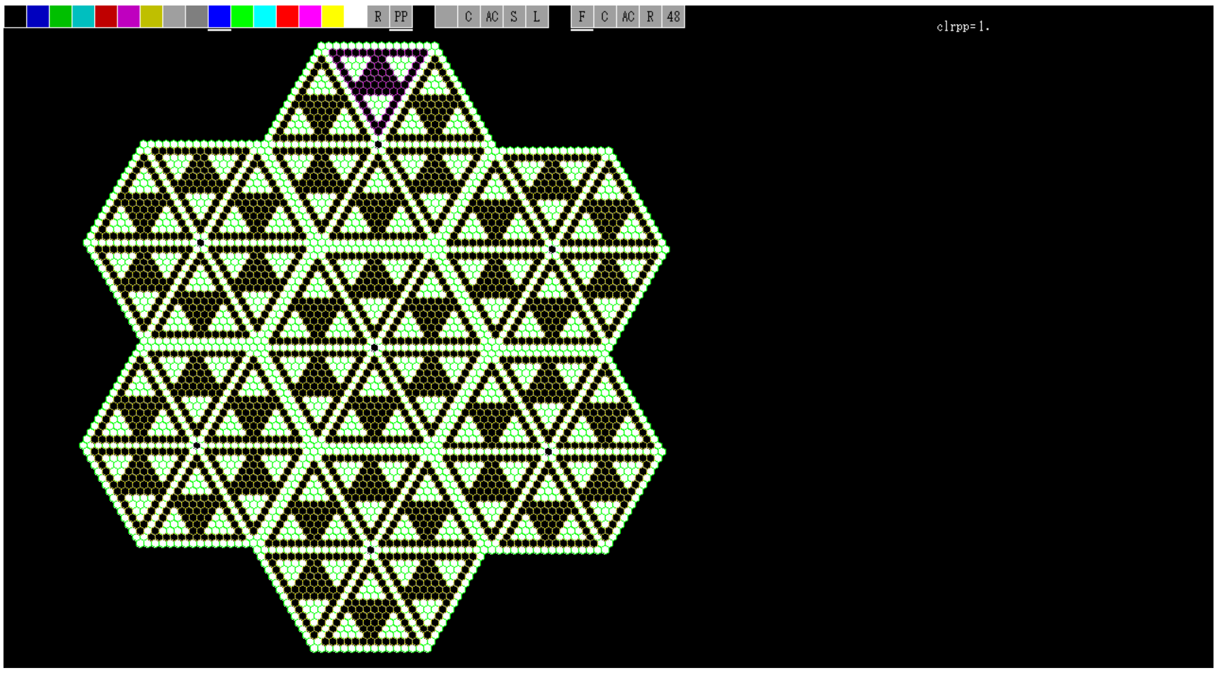

Figure 26

Copying the figure with painting detection method, if we do painting, C′ in Figure 27 is got.

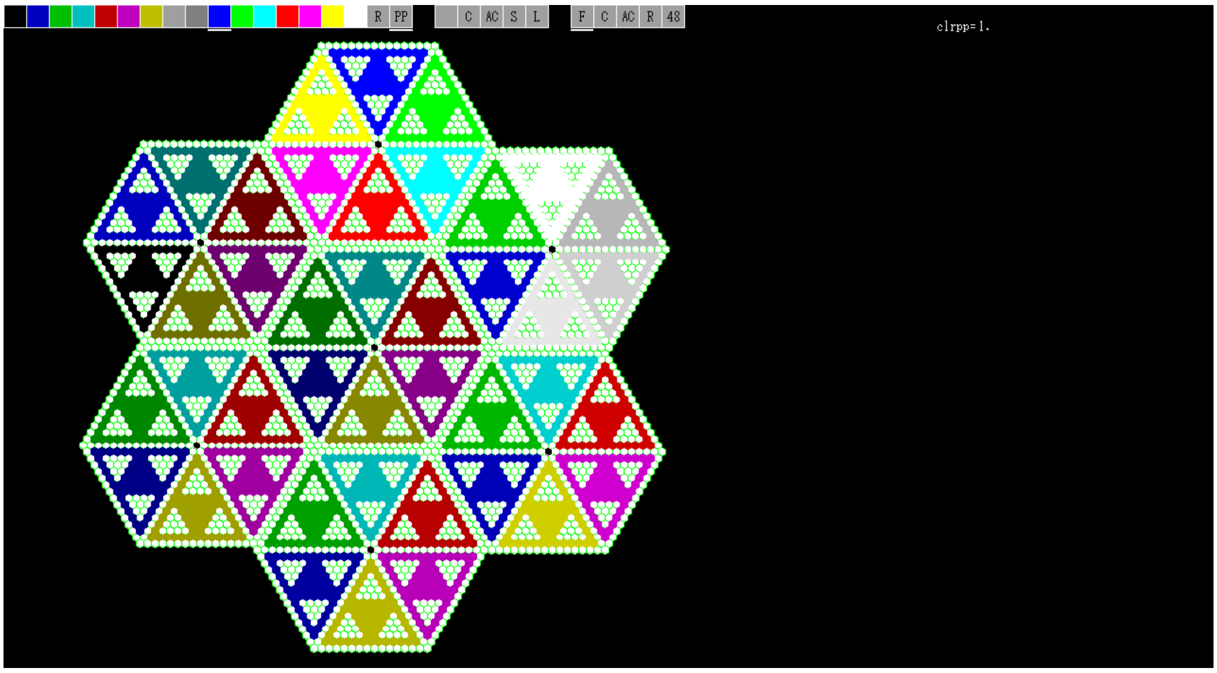

Figure 27

8. Array

If we make figures regularly by handiwork, two arrays are needed. Figure 2 shows array for display and array for painting support. The number in the figure is color code.

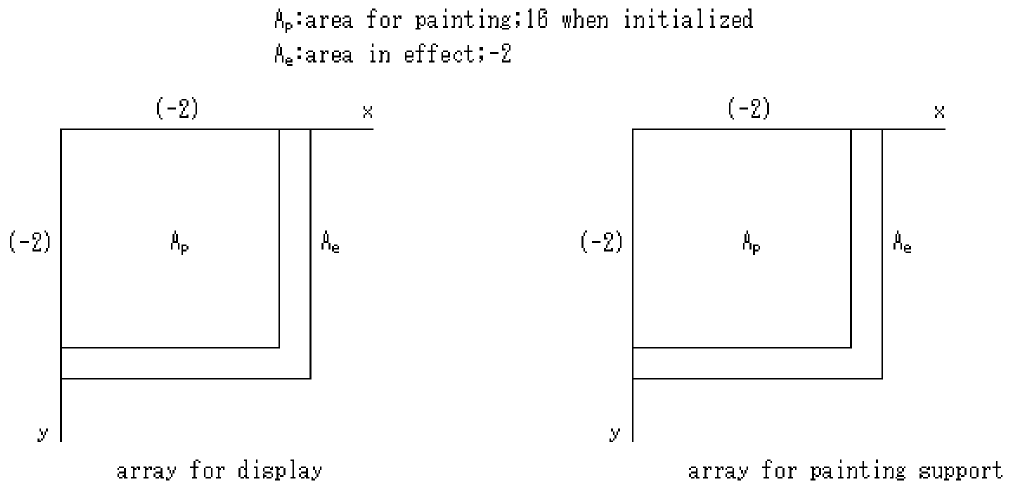

Figure 28

Area which is allocated must be a little larger than area for painting.

Hole A～F in Figure 10, hole A～D in Figure 13 in the 6th and cut part of the left figure in Figure 14 in the 8th, too, are initialized to -2. (-2) means -2 is returned if an array subscript is minus value.

Array for painting support is used when copied figures are made of a seed figure with painting detection method. Color code of a seed figure in array for painting support is -1.

9. Application

Parallel shift method is used only in orthogonal coordinate system. Painting number method is used only for copy of one pixel. Painting detection method is used in all cases. We show the possibility of application of these methods to infinite plane in the following table. The inside of the brackets is the possibility of application on painting in different colors.

Table 1 infinite plane(square pixel)

	one pixel	domain A	domain B	domain C
parallel shift	○(○)	○(○)	○(○)	○(○)
painting number	○(○)	○(×)	○(×)	○(×)
painting detection	○(○)	○(○)	○(○)	○(○)

Table 2 infinite plane(hexagonal pixel)

	one pixel	domain A	domain B	domain C
parallel shift	×	×	×	×
painting number	○(○)	○(△)	○(△)	○(△)
painting detection	○(○)	○(○)	○(○)	○(○)

Domain A, B, C are as follows:

- domain A : C+S, C+0, 0+S
- domain B : S′
- domain C : C′

In painting number method, a copied figure is disturbed if painting of domain A, B, C is in different colors. Figure 29, 30 are the examples. Consequently, painting number method is used chiefly for copy of seed point in painting detection method.

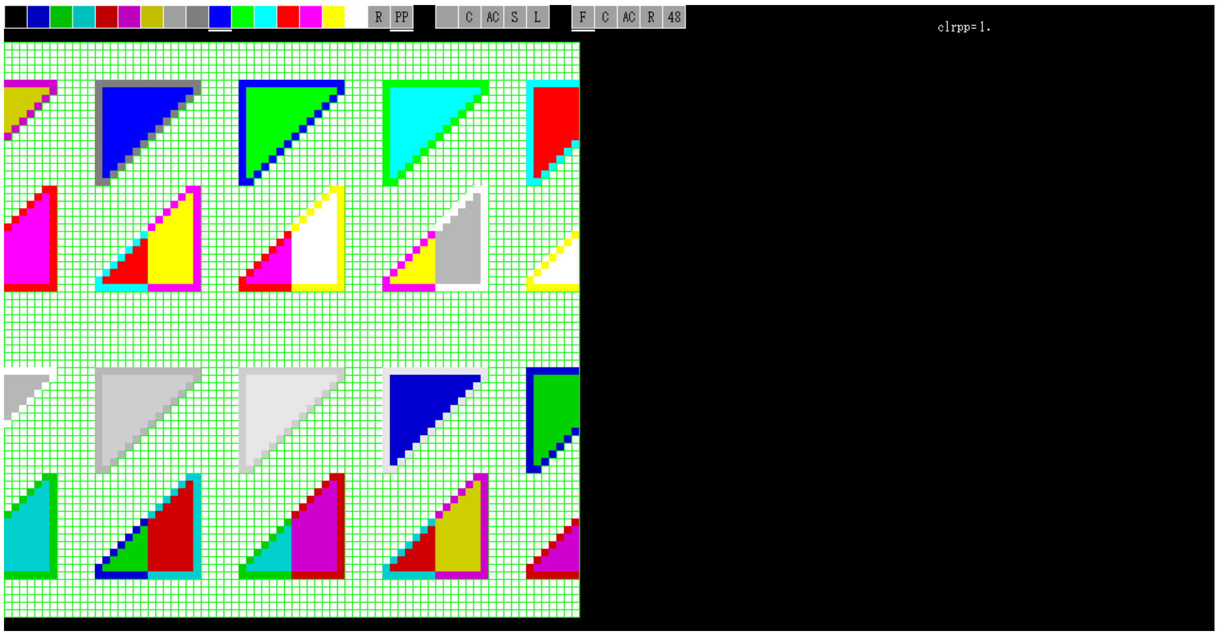

Figure 29

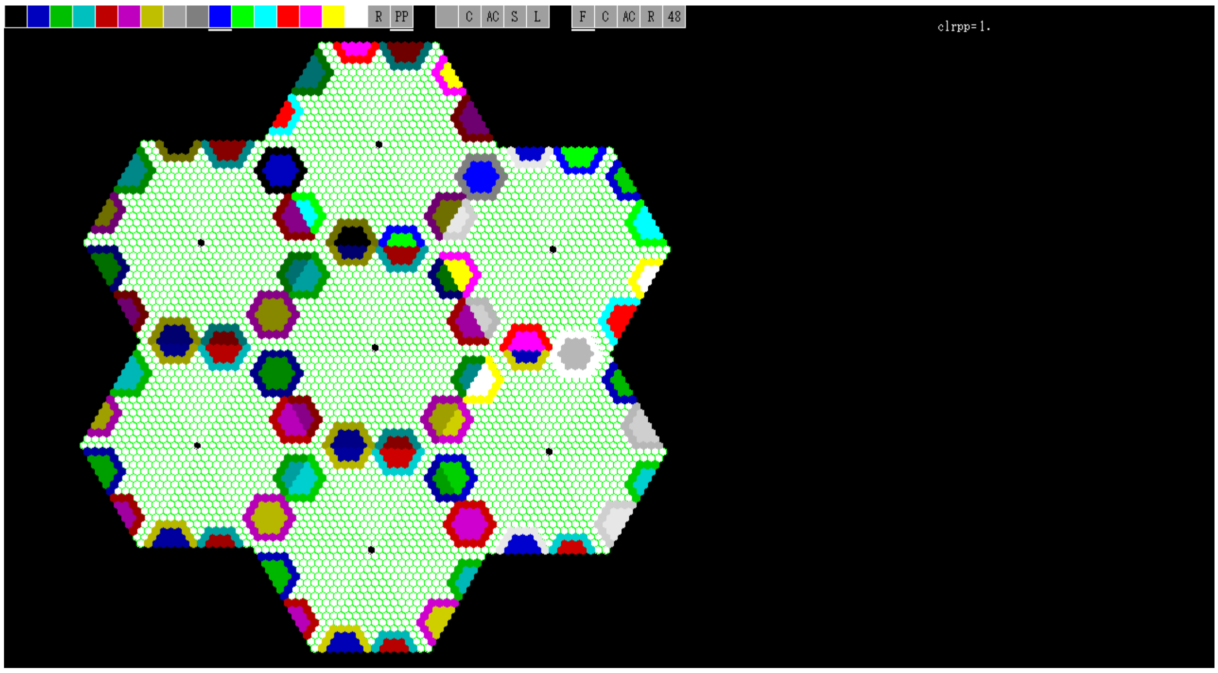

Figure 30

We show the possibility of application of them to polyhedron in the following table.

Table 3 polyhedron

	one pixel	domain A	domain B	domain C
parallel shift	×	×	×	×
painting number	○(○)	○( )	○( )	○( )
painting detection	○(○)	○(○)	○(○)	○(○)

10. Modification of the past portion

Modify the sentences in void set_vals(int j) in the 6th.

・wrong :

```
if(pixel[X][Y]==0) getpixel_(X_,Y_,X,Y);
    enX[j]=X;enY[j]=Y;enX_[j]=X_;enY_[j]=Y_;
```

・right：

```
if(Y<0 || X<0){ getpixel_(X_,Y_,X,Y);
    enX[j]=X;enY[j]=Y;enX_[j]=X_;enY_[j]=Y_;
}
```

In some compilers, pixel[0][-1]==0 is possible.
  Add the following to prototype declaration.

```
/* the 7th */
void pp_(int,int,int);

/* the 8th */
void setstccolor(int);
void paint(char,int,int,int,int,int);
void C_and_S(char,int,int);
void arrayreset_(void);
void search(int,int,int);
void setcsrcolor(int);
void imm_check(void);
void imm_pause(void);
void imm_close(void);
void breaks(char);
#if WX==0
void WM_func_CHAR(WPARAM);
void WM_funcIME_CHAR(WPARAM);
void WM_funcIME_STARTCOMPOSITION(void);
void WM_funcIME_COMPOSITION(LPARAM);
void WM_funcIME_ENDCOMPOSITION(void);
#else
void WM_func_CHAR(unsigned char);
void WM_funcIME_CHAR(unsigned char *);
#endif
void csr(void);
void left_keydowns_dialog(void);
void backspace_dialog(void);
void text_end_dialog(void);
void page_down_dialog(void);
void page_up_dialog(void);
void trim_dialog(void);
void restore_dialog(void);
void clear_dialog(char);
void BitBlt_dialog(char);
void tailcheck_dialog(void);
void within_linemax_dialog(void);
void csr_row_home_dialog(void);
void csr_row_end_dialog(void);
void csr_right_dialog(void);
void fsave(void);
void fload(void);
void restore_edge(void);

/* the 9th */
void setstccolor(int);
void paint(char,int,int,int,int,int);
void C_and_S(char,int,int);
void arrayreset_(void);
void search_bdr(int,int);
void search_i(int,int,int);
void setcsrcolor(int);
void imm_check(void);
void imm_pause(void);
void imm_close(void);
void breaks(char);
#if WX==0
void WM_func_CHAR(WPARAM);
void WM_funcIME_CHAR(WPARAM);
void WM_funcIME_STARTCOMPOSITION(void);
void WM_funcIME_COMPOSITION(LPARAM);
void WM_funcIME_ENDCOMPOSITION(void);
#else
void WM_func_CHAR(unsigned char);
void WM_funcIME_CHAR(unsigned char *);
#endif
void csr(void);
void left_keydowns_dialog(void);
void backspace_dialog(void);
void text_end_dialog(void);
void page_down_dialog(void);
void page_up_dialog(void);
void trim_dialog(void);
void restore_dialog(void);
void clear_dialog(char);
void BitBlt_dialog(char);
void tailcheck_dialog(void);
void within_linemax_dialog(void);
void csr_row_home_dialog(void);
void csr_row_end_dialog(void);
void csr_right_dialog(void);
void fsave(char *);
void fload(char *);
void restore_edge(void);
```

11. Concrete example

If program is executed, painting in which coordinates are memorized into painting number array begins. If the painting ends, area for painting is initialized. The following is painting procedure.

・Click F and PP with left button.
・Make a 8-connected closed curve with drag while pressing of left button.
・Click a pixel inside the closed curve with right button while pressing Ctrl key and Shift key. C+S is got.
・Click light green of palette with left button.
・Click a blue pixel in the inside of the closed curve which was painted with right button. S′ is got.
・Click F, left AC and F in order with left button.
・Make a 8-connected closed curve with drag while pressing of left button.
・Click a pixel inside the closed curve with right button while pressing Ctrl key. C+0 is got.
・Click F, left AC and F in order with left button.
・Make a 8-connected closed curve with drag while pressing of left button.
・Click a pixel inside the closed curve with right button while pressing Shift key. 0+S is got.
・Click F, left AC and F in order with left button.
・Make a 8-connected domain with drag while pressing of left button.
・Click a pixel in the domain with right button while pressing Ctrl key. C′ is got.

The following is operation when marker is under F.

・left click：addition of one pixel
・right C：deletion of one pixel
・right AC：deletion of all pixels

The following is operation after painting.

・right R：restoration of a state before painting

A state before painting is saved into tmp.bin. We restore a state before painting doing a load out of tmp.bin.
The following is operation when marker is not under F.
・left click：addition of one pixel
・left C：deletion of one pixel
・left AC：deletion of all pixels
・S：save into a file
・L：load out of a file

The following is symbolic constant.

・`#define HEX_or_SQR`：choice of pixel(0:hexagonal pixel, 1:square pixel)
・`#define Lhex`：number of hexagonal pixels of a side of D_1
・`#define Lsqr`：(number of square pixels between two seed points)/2

- `#define PIXSIZE`：size of pixel
- `#define XRESO`：horizontal size of window
- `#define YRESO`：vertical size of window
- `#define X0`：position in the direction of x of graphics
- `#define PHASE`：phase(0:parallelism, 1:antiparallelism, 2:oblique antiparallelism)

****

セルラーオートマトングラフィクス (10)

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. 無限平面 (正方形ピクセル)

　図 1 は無限平面を敷詰めるための正方形ピクセルを用いる領域です。

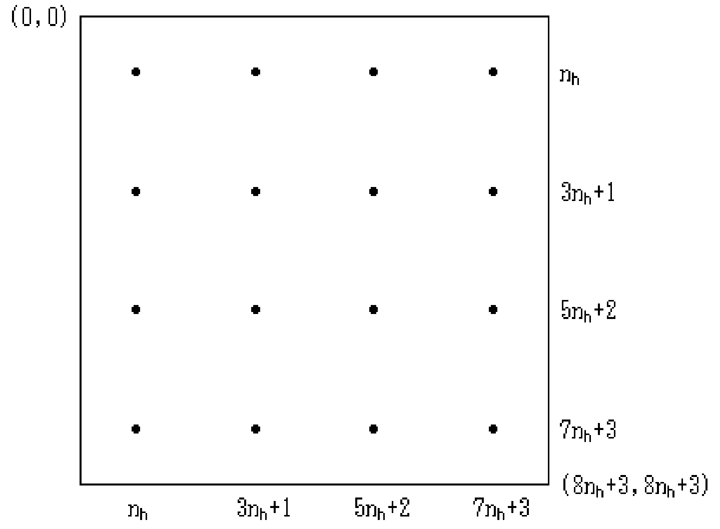

図 1

　図における点はシードポイントです。シードポイントの座標は、n_s を (シードポイント間の正方形ピクセル数)/2 として、以下のようになっています。

・$(n_\mathrm{s},\, n_\mathrm{s}),\, (3n_\mathrm{s}+1,\, n_\mathrm{s}),\, (5n_\mathrm{s}+2,\, n_\mathrm{s}),\, (7n_\mathrm{s}+3,\, n_\mathrm{s})$
・$(n_\mathrm{s},\, 3n_\mathrm{s}+1),\, (3n_\mathrm{s}+1,\, 3n_\mathrm{s}+1),\, (5n_\mathrm{s}+2,\, 3n_\mathrm{s}+1),\, (7n_\mathrm{s}+3,\, 3n_\mathrm{s}+1)$
・$(n_\mathrm{s},\, 5n_\mathrm{s}+2),\, (3n_\mathrm{s}+1,\, 5n_\mathrm{s}+2),\, (5n_\mathrm{s}+2,\, 5n_\mathrm{s}+2),\, (7n_\mathrm{s}+3,\, 5n_\mathrm{s}+2)$
・$(n_\mathrm{s},\, 7n_\mathrm{s}+3),\, (3n_\mathrm{s}+1,\, 7n_\mathrm{s}+3),\, (5n_\mathrm{s}+2,\, 7n_\mathrm{s}+3),\, (7n_\mathrm{s}+3,\, 7n_\mathrm{s}+3)$
・$n_\mathrm{s} > 0$

プログラムでは n_s の代わりに Lsqr を用いています。
　シードポイントの次の塗点は座標の変化量を例えば以下のように選んで決めます。

・$(n_\mathrm{s},\, n_\mathrm{s}),\, (3n_\mathrm{s}+1,\, n_\mathrm{s}),\, (5n_\mathrm{s}+2,\, n_\mathrm{s}),\, (7n_\mathrm{s}+3,\, n_\mathrm{s})$：$\Delta x = 1$
・$(n_\mathrm{s},\, 3n_\mathrm{s}+1),\, (3n_\mathrm{s}+1,\, 3n_\mathrm{s}+1),\, (5n_\mathrm{s}+2,\, 3n_\mathrm{s}+1),\, (7n_\mathrm{s}+3,\, 3n_\mathrm{s}+1)$：$\Delta x = 1$
・$(n_\mathrm{s},\, 5n_\mathrm{s}+2),\, (3n_\mathrm{s}+1,\, 5n_\mathrm{s}+2),\, (5n_\mathrm{s}+2,\, 5n_\mathrm{s}+2),\, (7n_\mathrm{s}+3,\, 5n_\mathrm{s}+2)$：$\Delta x = 1$
・$(n_\mathrm{s},\, 7n_\mathrm{s}+3),\, (3n_\mathrm{s}+1,\, 7n_\mathrm{s}+3),\, (5n_\mathrm{s}+2,\, 7n_\mathrm{s}+3),\, (7n_\mathrm{s}+3,\, 7n_\mathrm{s}+3)$：$\Delta x = 1$
・$\Delta y = 0$

・$(n_\mathrm{s},\, n_\mathrm{s}),\, (3n_\mathrm{s}+1,\, n_\mathrm{s}),\, (5n_\mathrm{s}+2,\, n_\mathrm{s}),\, (7n_\mathrm{s}+3,\, n_\mathrm{s})$：$\Delta x = 1$

・$(n_s,\ 3n_s+1),\ (3n_s+1,\ 3n_s+1),\ (5n_s+2,\ 3n_s+1),\ (7n_s+3,\ 3n_s+1)$：$\Delta x=-1$
・$(n_s,\ 5n_s+2),\ (3n_s+1,\ 5n_s+2),\ (5n_s+2,\ 5n_s+2),\ (7n_s+3,\ 5n_s+2)$：$\Delta x=1$
・$(n_s,\ 7n_s+3),\ (3n_s+1,\ 7n_s+3),\ (5n_s+2,\ 7n_s+3),\ (7n_s+3,\ 7n_s+3)$：$\Delta x=-1$
・$\Delta y=0$

または

・$(n_s,\ n_s)$：$\Delta x=1$, $(3n_s+1,\ n_s)$：$\Delta x=-1$, $(5n_s+2,\ n_s)$：$\Delta x=1$,
$(7n_s+3,\ n_s)$：$\Delta x=-1$
・$(n_s,\ 3n_s+1)$：$\Delta x=-1$, $(3n_s+1,\ 3n_s+1)$：$\Delta x=1$, $(5n_s+2,\ 3n_s+1)$：$\Delta x=-1$,
$(7n_s+3,\ 3n_s+1)$：$\Delta x=1$
・$(n_s,\ 5n_s+2)$：$\Delta x=1$, $(3n_s+1,\ 5n_s+2)$：$\Delta x=-1$, $(5n_s+2,\ 5n_s+2)$：$\Delta x=1$,
$(7n_s+3,\ 5n_s+2)$：$\Delta x=-1$
・$(n_s,\ 7n_s+3)$：$\Delta x=-1$, $(3n_s+1,\ 7n_s+3)$：$\Delta x=1$, $(5n_s+2,\ 7n_s+3)$：$\Delta x=-1$,
$(7n_s+3,\ 7n_s+3)$：$\Delta x=1$
・$\Delta y=0$

これらは各々平行、反平行、斜め反平行と称されます。
　塗りつぶしアルゴリズムは第 6 回の図 5 の CW、図 6 の CCW を用います。塗点は領域からはみ出ると周期的境界条件によってジャンプを行います。
　図 2 は描画例です。

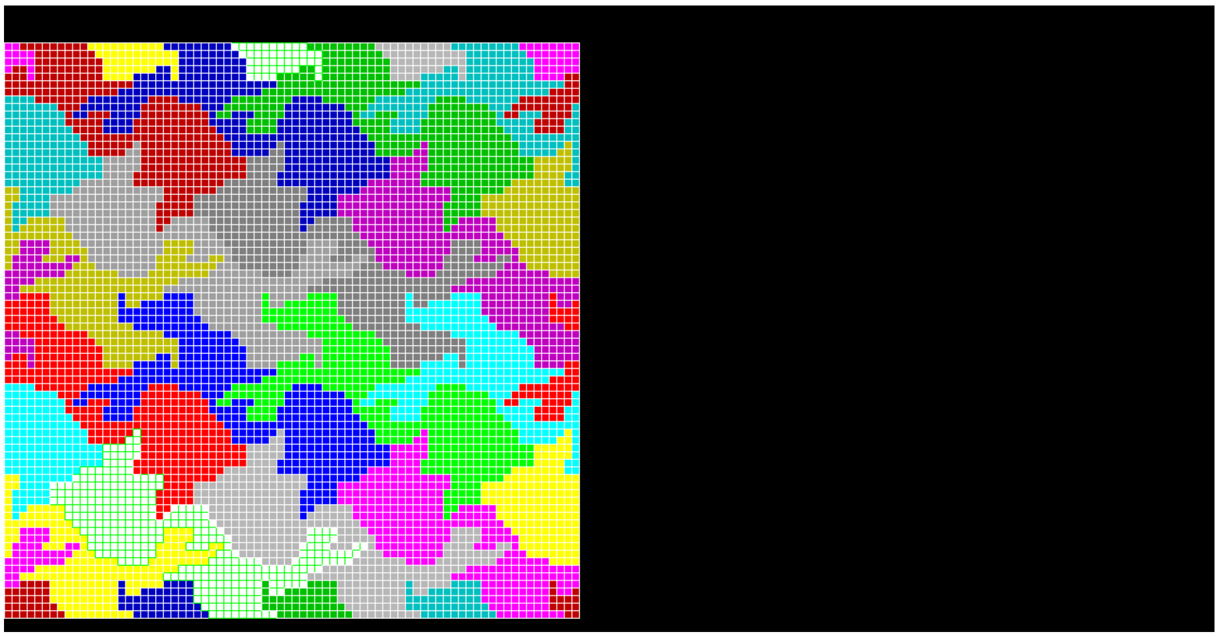
図 2

2. 無限平面 (正六角形ピクセル)
　図 3 は無限平面を敷き詰めるための正六角形ピクセルを用いる領域 D_1 です。図における黒いピクセルは D_1 の Dead Pixel です。

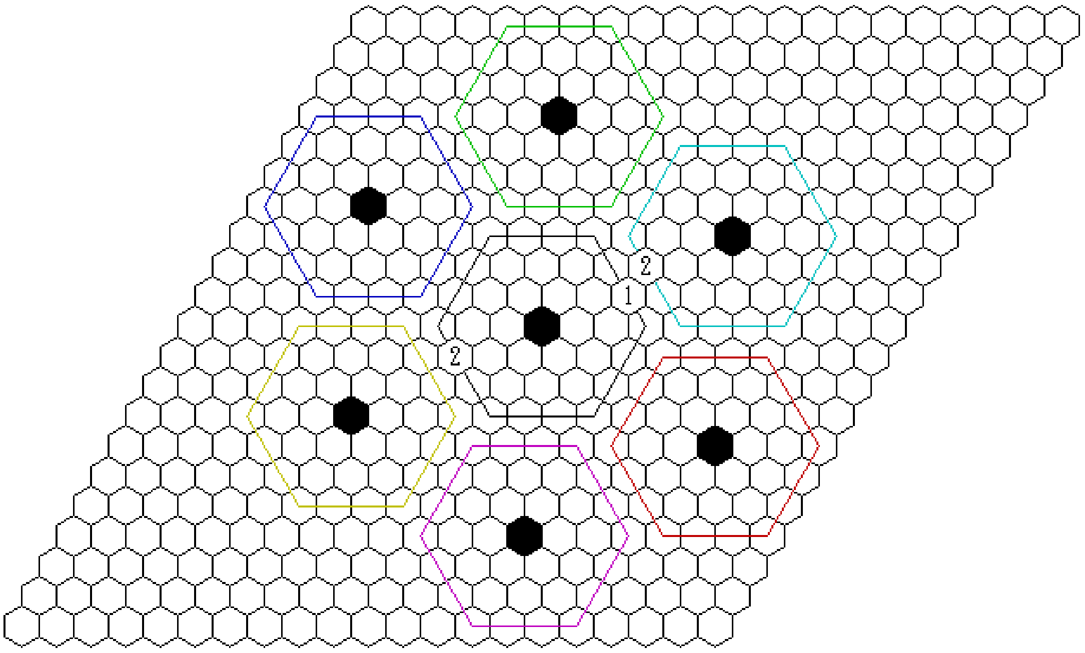

図 3

図 4 は D_1 を 7 個組み合わせた領域 D_7 です。

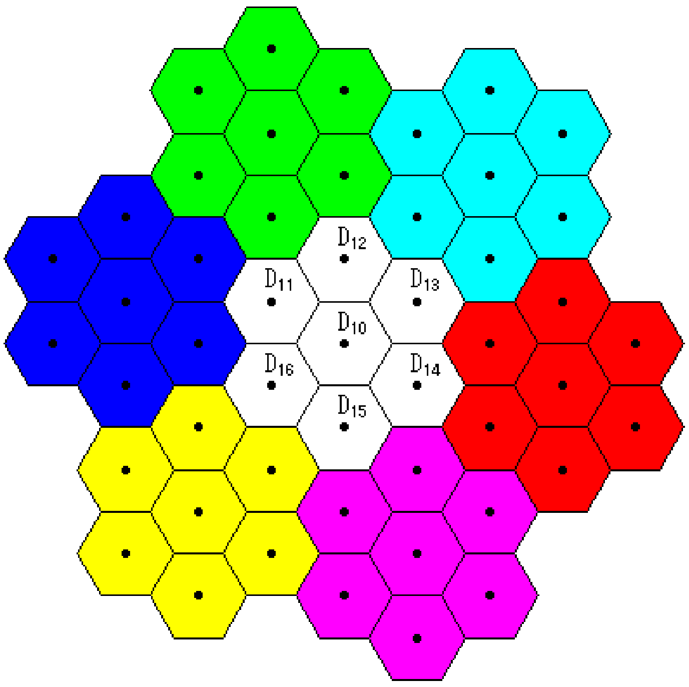

図 4

図における点は D_1 の Dead Pixel です。Dead Pixel の座標は、n_h を D_1 の辺のピクセル数として、以下のようになっています。

・D_{10}：$(3(n_\mathrm{h}-1)+2\equiv X_0,\ 3(n_\mathrm{h}-1)+2\equiv Y_0)$ または $(3(n_\mathrm{h}-1)+1\equiv X_0,\ 3(n_\mathrm{h}-1)+1\equiv Y_0)$
・D_{11}：$(X_0-2(n_\mathrm{h}-1)-1,\ Y_0-(n_\mathrm{h}-1)-1)$
・D_{12}：$(X_0-(n_\mathrm{h}-1),\ Y_0-2(n_\mathrm{h}-1)-1)$
・D_{13}：$(X_0+n_\mathrm{h},\ Y_0-(n_\mathrm{h}-1))$
・D_{14}：$(X_0+2(n_\mathrm{h}-1)+1,\ Y_0+n_\mathrm{h})$
・D_{15}：$(X_0+(n_\mathrm{h}-1),\ Y_0+2(n_\mathrm{h}-1)+1)$
・D_{16}：$(X_0-n_\mathrm{h},\ Y_0+(n_\mathrm{h}-1))$

D_{10} の頂点の座標は第四回の図 4 の記号を用いて以下のようになっています。

・v0：$(X_0-(n_\mathrm{h}-1),\, Y_0-(n_\mathrm{h}-1))$
・v1：$(X_0,\, Y_0-(n_\mathrm{h}-1))$
・v2：$(X_0+(n_\mathrm{h}-1),\, Y_0)$
・v3：$(X_0+(n_\mathrm{h}-1),\, Y_0+(n_\mathrm{h}-1))$
・v4：$(X_0,\, Y_0+(n_\mathrm{h}-1))$
・v5：$(X_0-(n_\mathrm{h}-1),\, Y_0)$

プログラムでは n_h の代わりに Lhex を用いています。
D_1、D_7 のピクセル数は、Dead Pixel を含めて

・D_1：$3{n_\mathrm{h}}^2-3n_\mathrm{h}+1 \equiv N_1$
・D_7：$7N_1$

　シードポイントは例えば図 5 のように Dead Pixel の周りに 6 個配置します。ダッシのついたピクセルはシードポイントの次の塗点です。シードポイント、シードポイントの次の塗点は 6 回対称の関係になるように選びます。

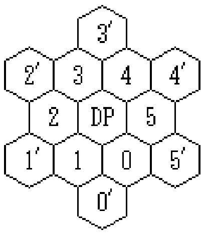

図 5

　塗りつぶしアルゴリズムは第 1 回の図 11 の CW、図 12 の CCW を用います。塗点は領域からはみ出ると周期的境界条件によってジャンプを行います。D_1 に関しては、図 3 において、例えば塗点が 1 から 2 に移動すると塗点はこの 2 から別の 2 にジャンプします。D_7 に関しては、図 4 において、図 3 における外側の D_1 の配置を参考にしてジャンプ先の D_1 を求めて下さい。
　図 6 は描画例です。

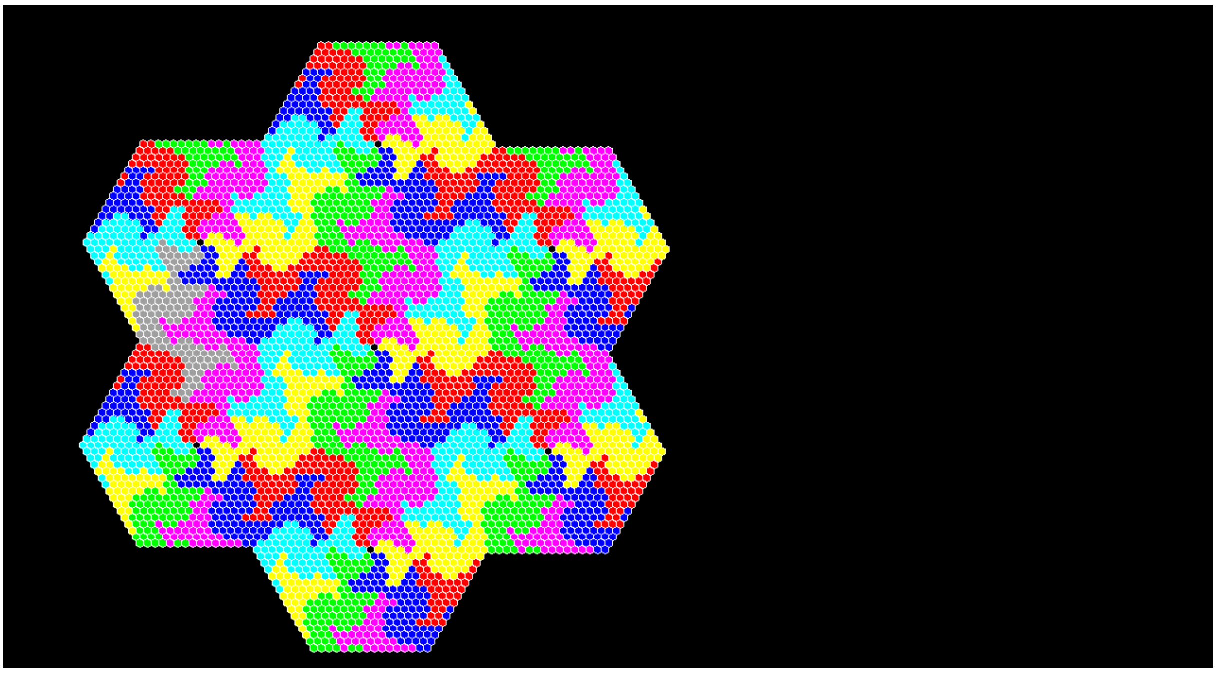
図 6

3. 手作業による敷き詰め

　CW、CCW の切り替えではなく、手作業で規則的な図形を作ることを考えます。セルラーオートマトングラフィクスでは以下のような領域を扱います。

・無限平面 (正方形ピクセル)
・無限平面 (正六角形ピクセル)
・多面体
・三次元空間
・多次元空間

第一回から第九回では多面体において CW、CCW を切り替えていました。

4. 無限平面 (正方形ピクセル)

　図 1 のような領域で手作業で図形を規則的に作る場合の最も簡単な方法は平行移動法です。図 7 においてある図形を手作業で、すなわちマウスを補助的に用いた塗りつぶしによって一つ作ったとします。これを種図形と称します。

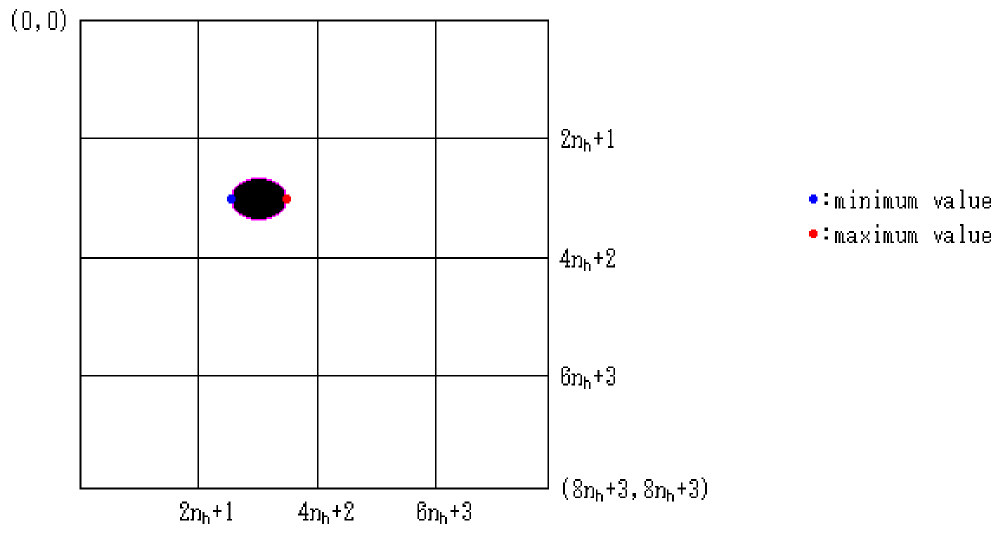

図 7

種図形を構成するピクセルを上下、左右にコピーします。種図形の x 方向の最小値と最大値を x_{min}、x_{max} とします。図では x_{min} が以下のような範囲にあります。

・$2n_{\mathrm{s}}+1 \leq x_{\mathrm{min}} < 4n_{\mathrm{s}}+2$

したがって、種図形のコピーの回数は左が 1、右が 2 です。もし、図 8 に示されるように x_{max} が以下のようになっていれば、2 回目のコピーでははみ出た部分を左端に移します。

・$4n_{\mathrm{s}}+2 \leq x_{\mathrm{max}}$

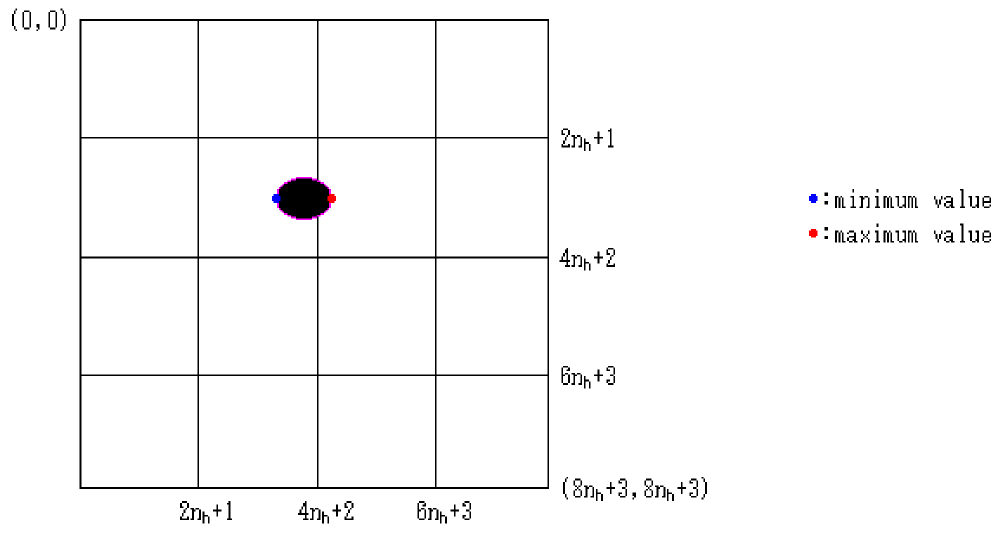

図 8

y 方向でも同様のコピーを行います。
平行移動法は直交座標だけを用いる場合に有効です。例えば、三次元空間では x 方向、y 方向、z 方向において上記のコピーを行います。

5. 無限平面 (正六角形ピクセル)

図 3 のような領域では塗番号法と塗りつぶし検知法を用います。塗番号法では、手作業による敷き詰めを行う前に、図 6 のような図形を CW だけまたは CCW だけで作成しながら塗りつぶしたピクセルの座標を塗番号毎に塗番号配列に記憶しておきます。塗番号は以下のようになっています。

・シードポイント (6 × 7 個)：0
・シードポイントの次の塗点 (6 × 7 個)：1
　　　　　・・・・・・・・・・・・・・・・・・・・・・・・・・
・最終塗点 (6 × 7 個)：一つの塗点に関する塗りつぶしたピクセルの数 −1

　図 9 において種図形をマウスを補助的に用いた塗りつぶしによって作ったとします。

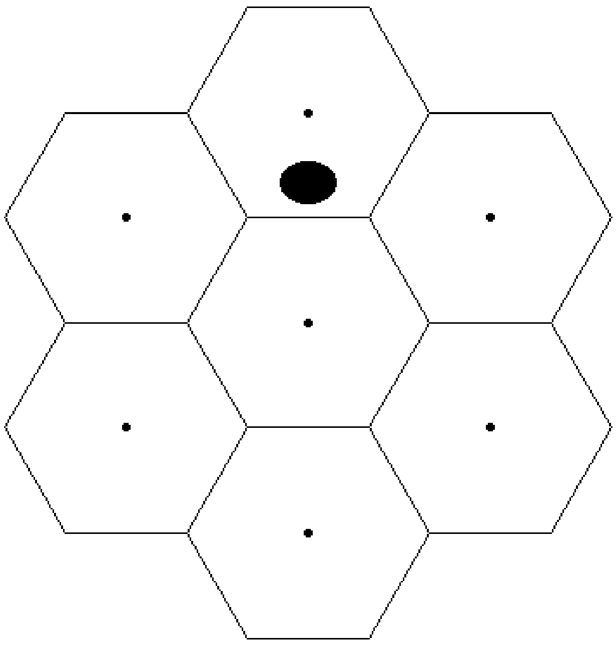
図 9

種図形を構成するあるピクセルの座標を塗番号配列から探し出し、まずそのピクセルを塗りつぶします。次にこのピクセルと塗番号が同じである塗番号配列の要素 (6 × 7 − 1 個) の座標を読み込んでピクセルを塗りつぶします。この作業を種図形を構成する全てのピクセルについて行います。
　塗りつぶし検知法では、図 6 のような図形を CW だけまたは CCW だけで作成しながら、もしある塗点が種図形を構成するピクセルの座標を検知した場合はその時の全ての塗点すなわちメンバーの座標を配列に記憶し、作成が終わった後でこの配列の要素の座標を読み込んで塗りつぶし用の領域を作成します。
　種図形は図 9 だけではなく、図 10 のように閉曲線のようなボーダーの場合もあります。この場合はこのボーダーを塗りつぶし検知法でコピーして塗りつぶし用の領域を作成します。

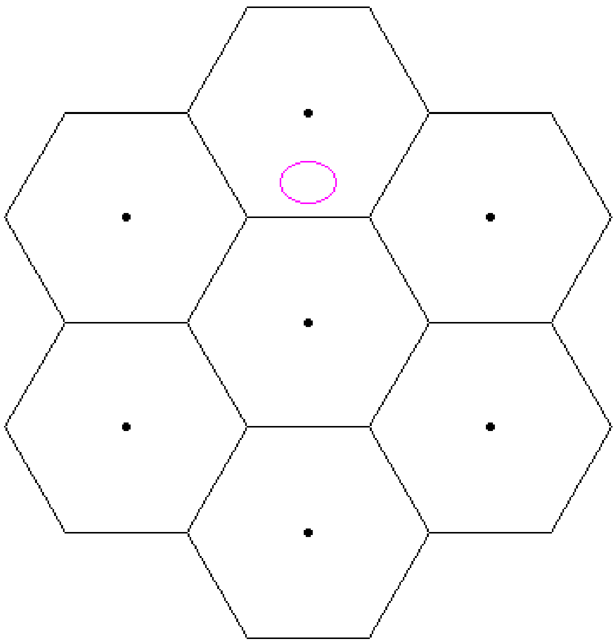
図 10

塗りつぶし用の領域を作成したら、シードポイントを塗番号法によって塗りつぶし用の領域にコピーし、塗りつぶしを行います。

6. 塗りつぶしアルゴリズム (正方形ピクセル)

正方形ピクセルの塗りつぶしでは連結数の異なる塗りつぶしアルゴリズムを用いることになります。図 11 はマウスで描いた 8 連結のボーダーです。このボーダーは閉じている必要があります。

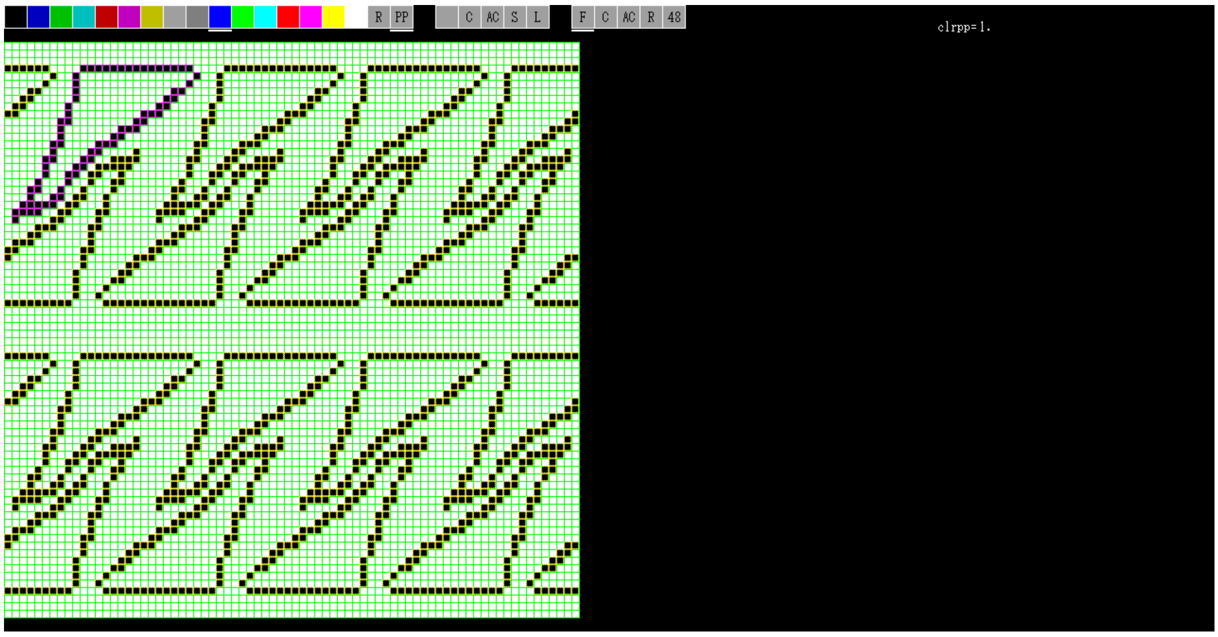

図 11

これを塗りつぶし検知法によってコピーしてボーダーの内部の塗りつぶしを行うと図 12 の C+S、図 13 の C+0、図 14 の 0+S が得られます。C はボーダー、S はその内部です。塗りつぶしアルゴリズムは図 15 を用います。C は S の塗りつぶしをする際にボーダーのピクセルを検知してこれを塗りつぶすことによって得られます。

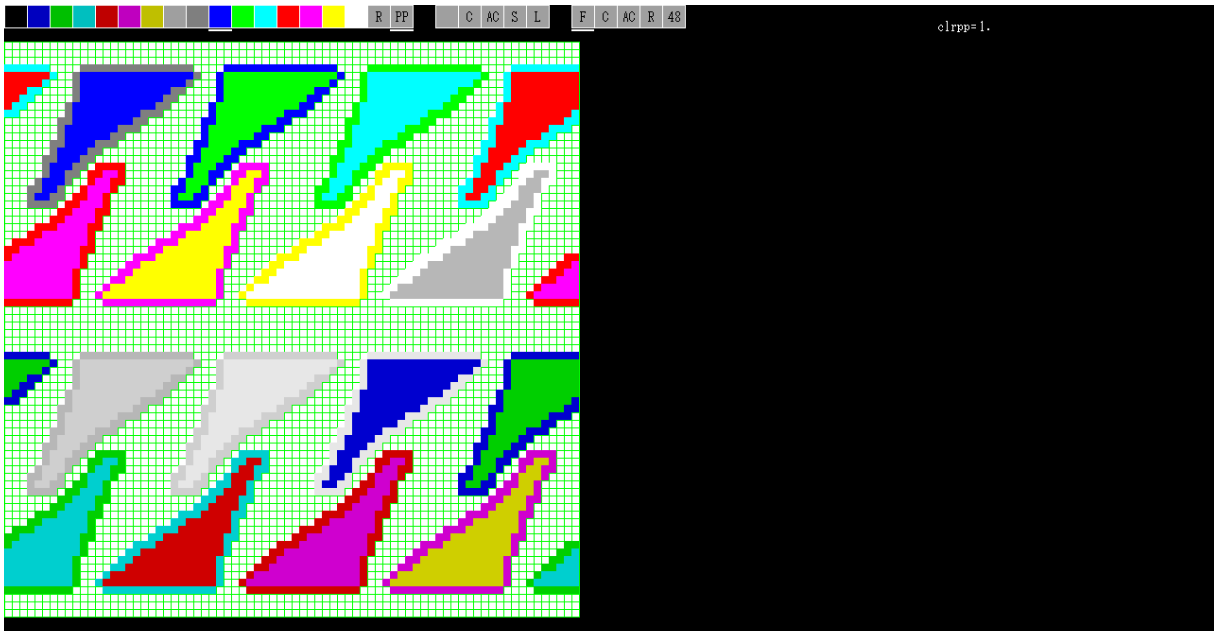

図 12

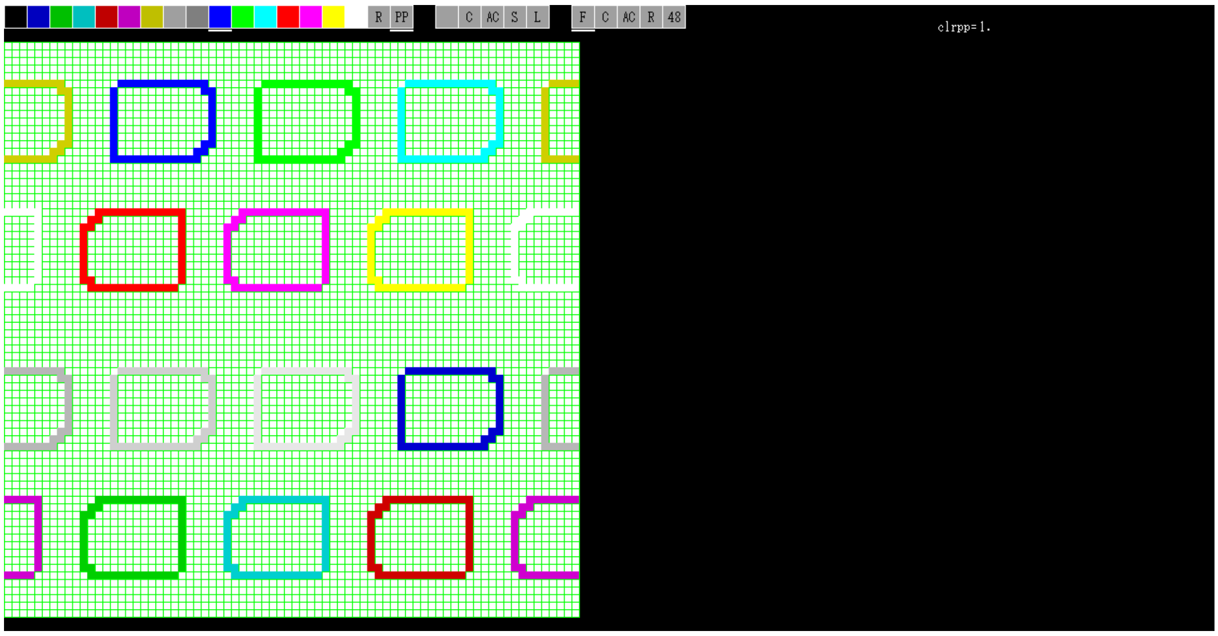

図 13

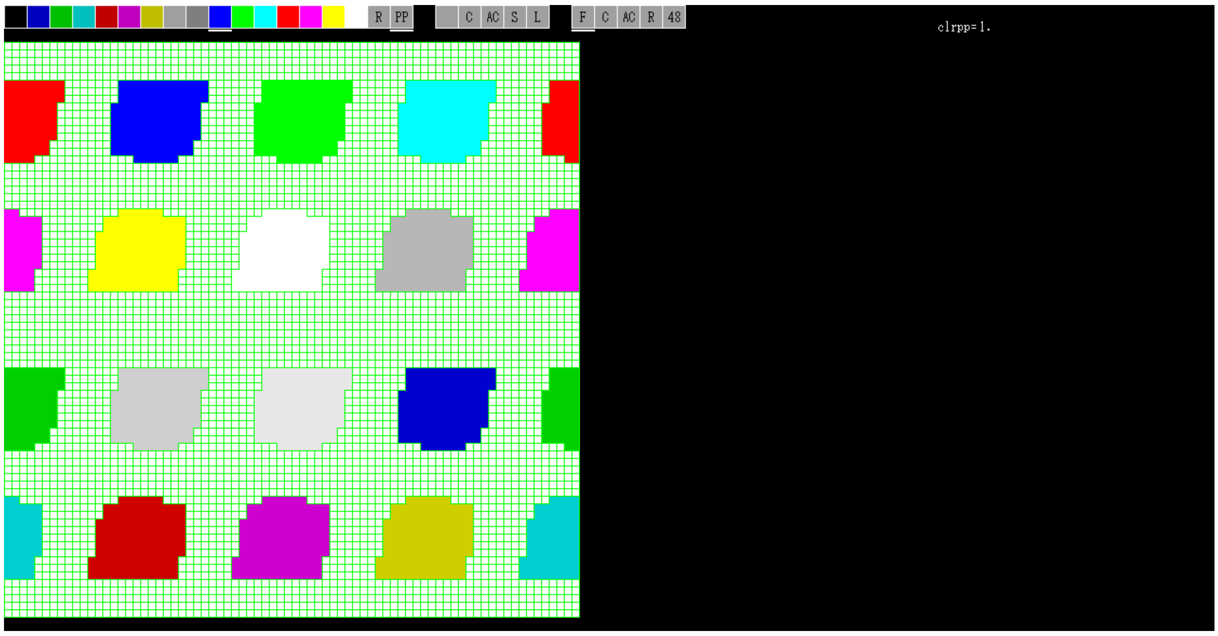

図 14

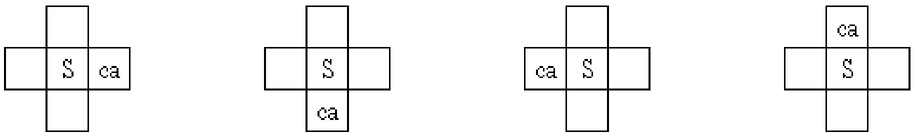

図 15

図 12 の C+S の S を種図形として、これを塗りつぶし検知法によってコピーして塗りつぶしを行うと図 16 の S′ が得られます。塗りつぶしアルゴリズムは図 15 を用います。

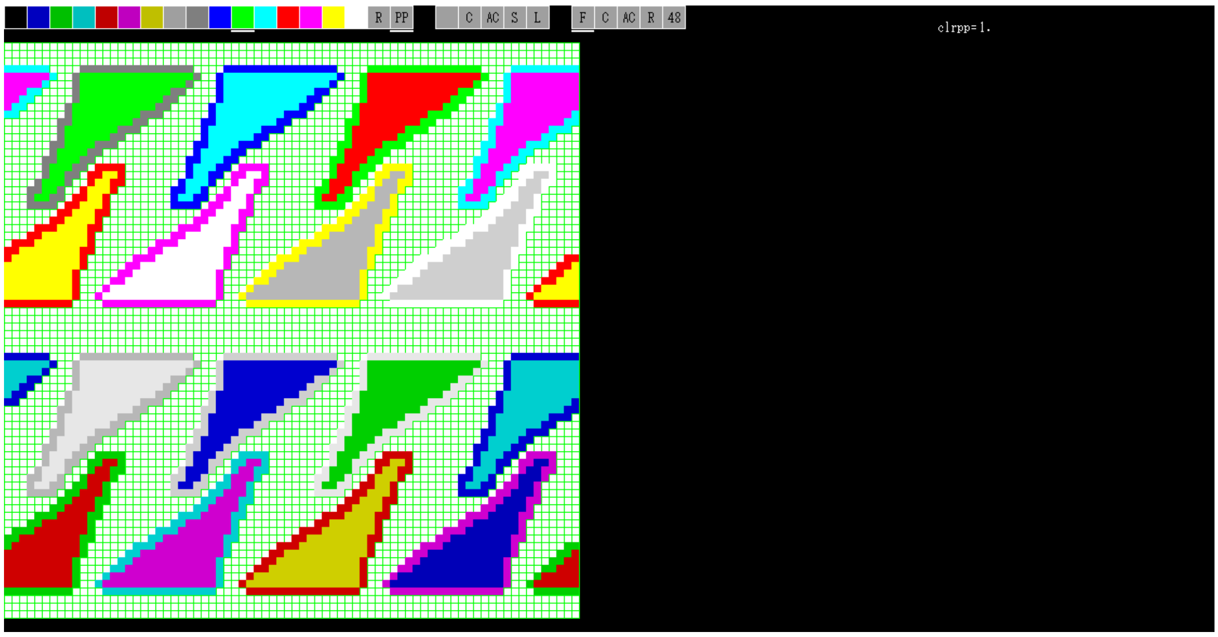

図 16

図 17 はマウスで描いた 8 連結の図形です。

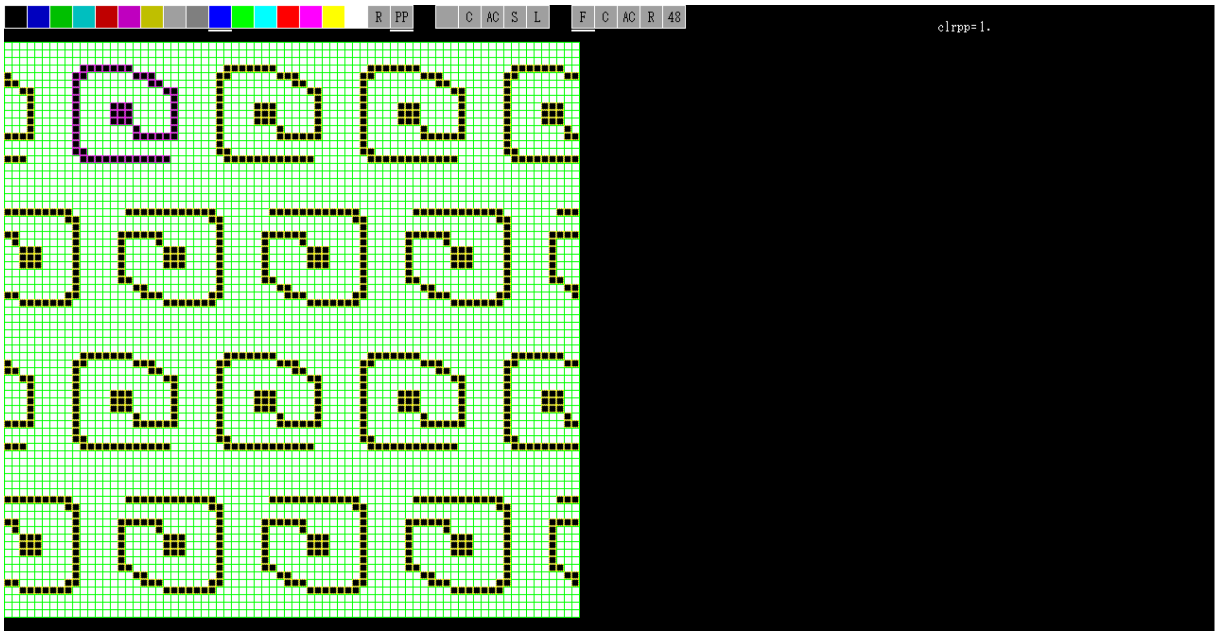

図 17

これを塗りつぶし検知法によってコピーして塗りつぶしを行うと図 18 の C′ が得られます。塗りつぶしアルゴリズムは図 19 を用います。

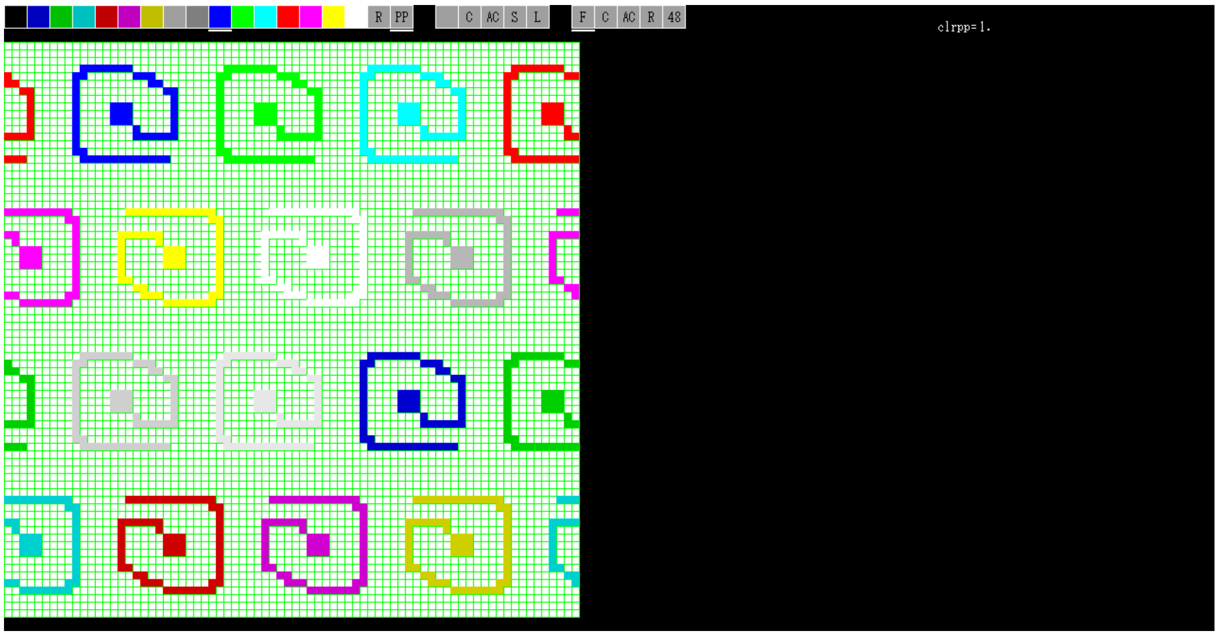

図 18

	S	ca

	S	
	ca	

ca	S	

	ca	
	S	

	S	
		ca

	S	
ca		

ca		
	S	

		ca
	S	

図 19

7. 塗りつぶしアルゴリズム (正六角形ピクセル)

正六角形ピクセルの塗りつぶしでは図 20 のアルゴリズムを用います。

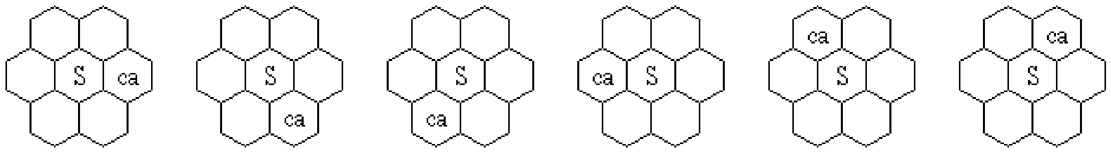

図 20

図 21 はマウスで描いた 6 連結のボーダーです。このボーダーは閉じている必要があります。

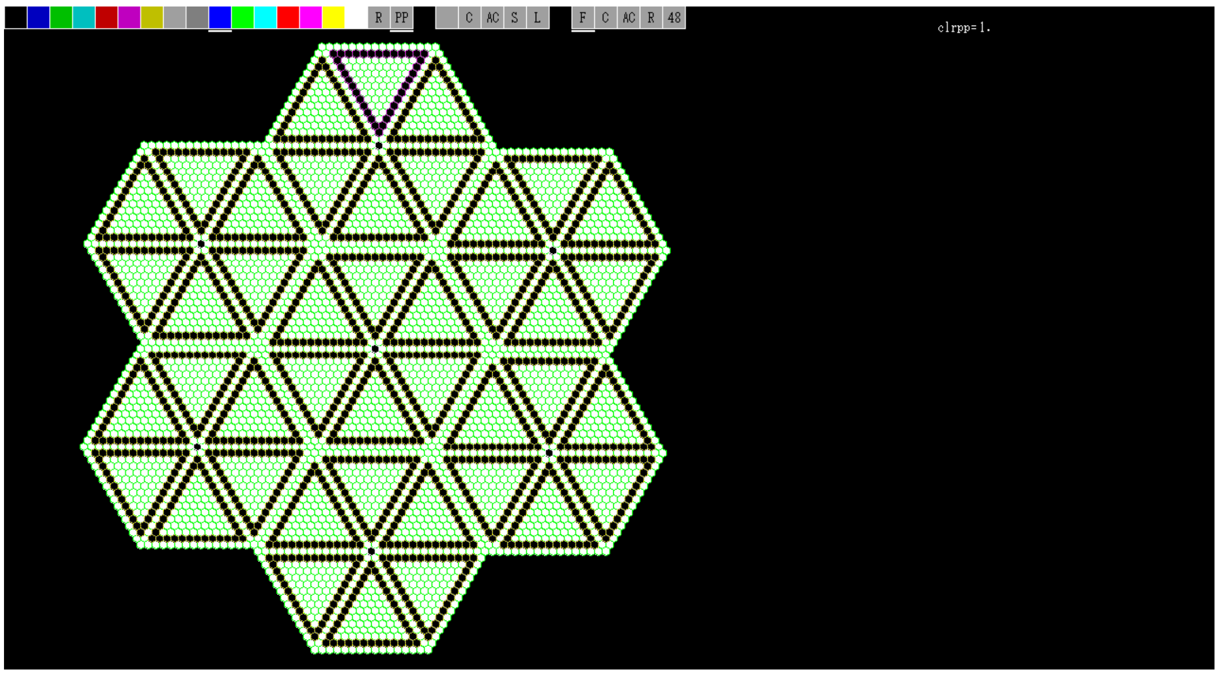

図 21

これを塗りつぶし検知法によってコピーしてボーダーの内部の塗りつぶしを行うと図 22 の C+S、図 23 の C+0、図 24 の 0+S が得られます。C はボーダー、S はその内部です。C は S の塗りつぶしをする際にボーダーのピクセルを検知してこれを塗りつぶすことによって得られます。

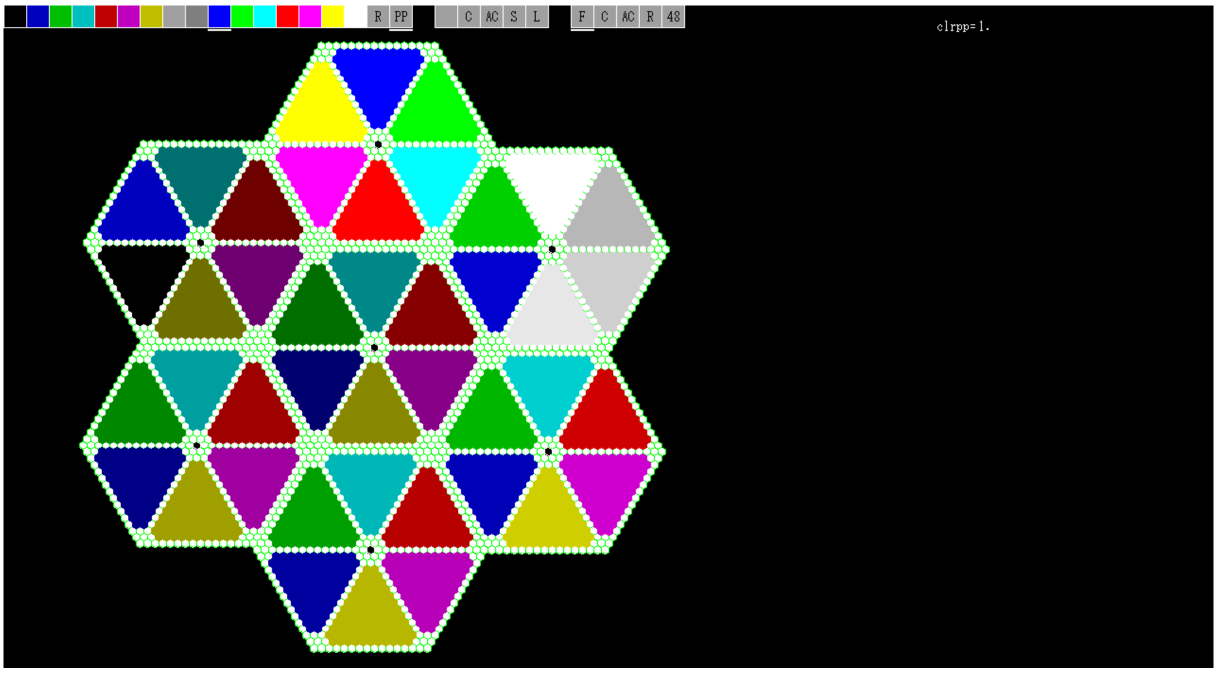

図 22

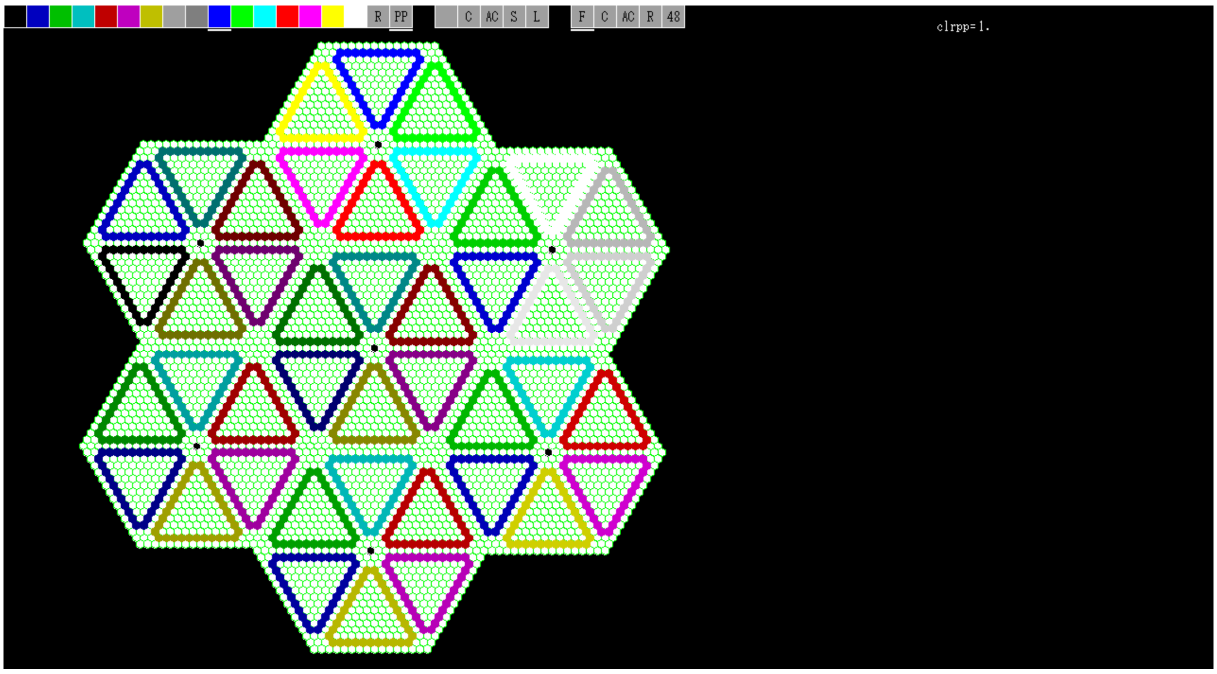

図 23

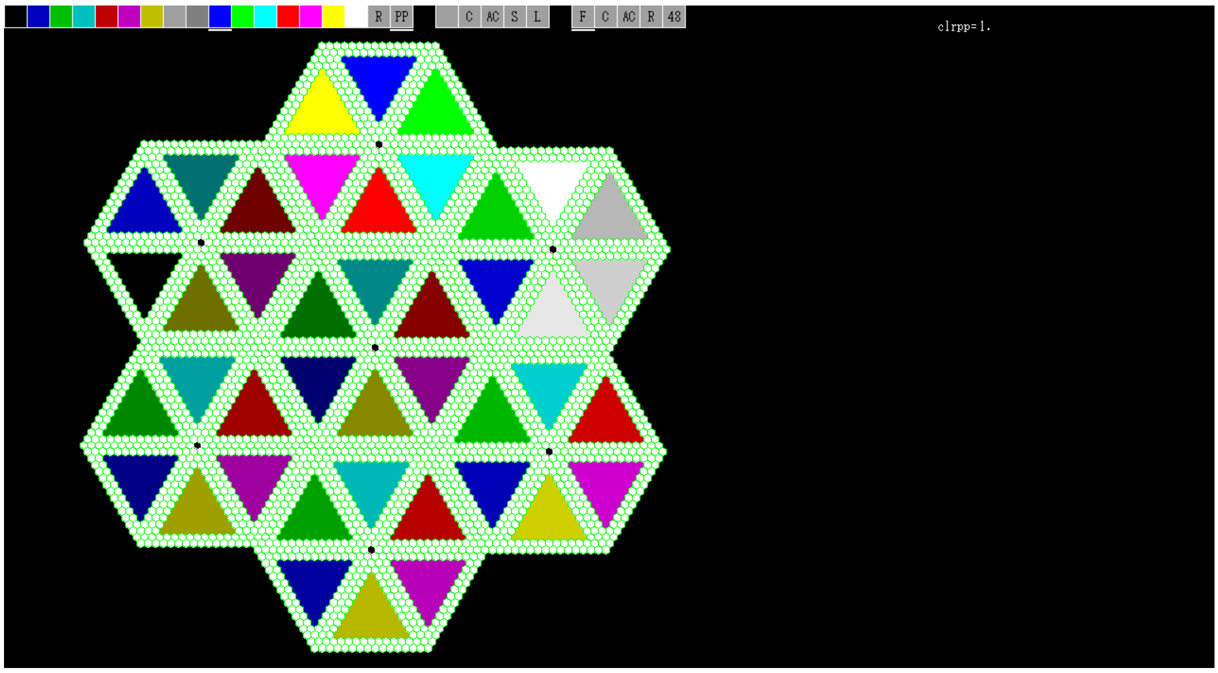

図 24

図 22 の C+S の S を種図形として、これを塗りつぶし検知法によってコピーして塗りつぶしを行うと図 25 の S′ が得られます。

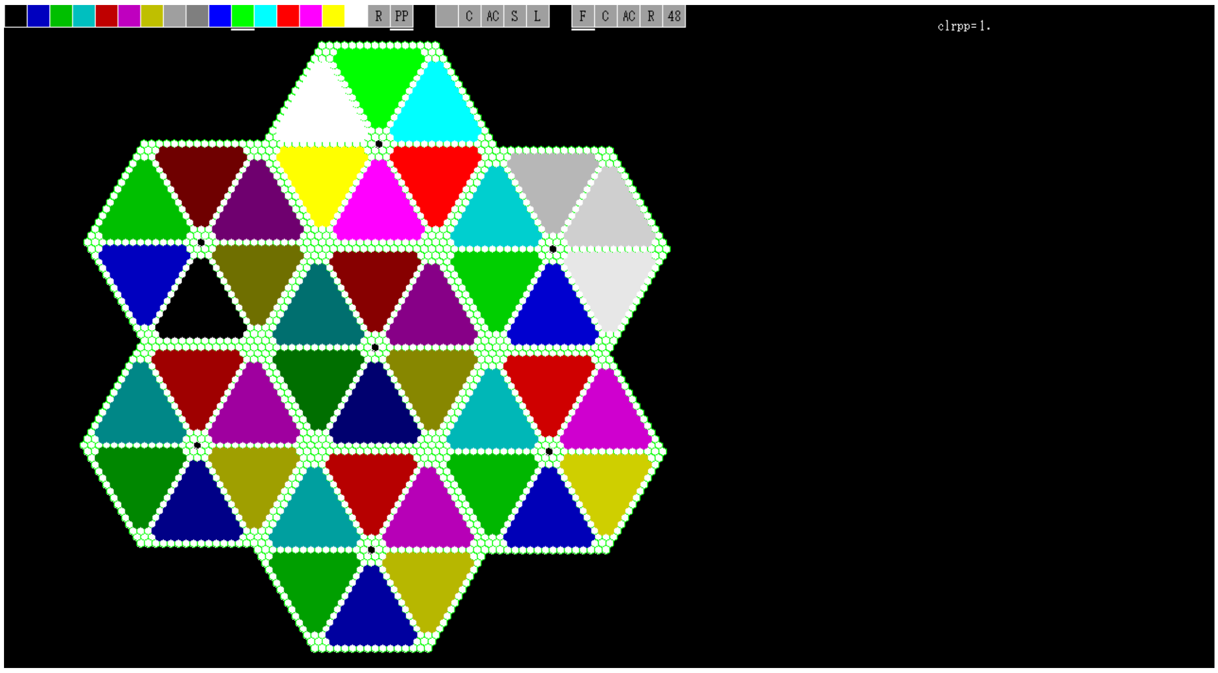

図 25

図 26 はマウスで描いた 6 連結の図形です。

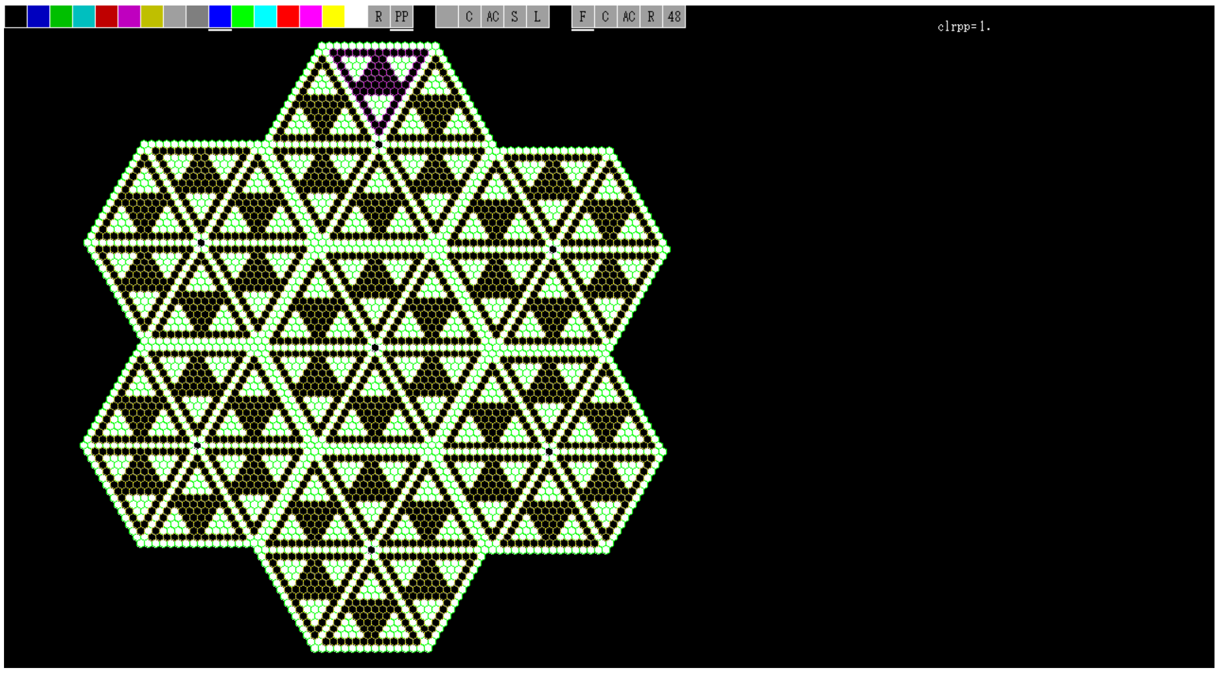

図 26

これを塗りつぶし検知法によってコピーして塗りつぶしを行うと図 27 の C′ が得られます。

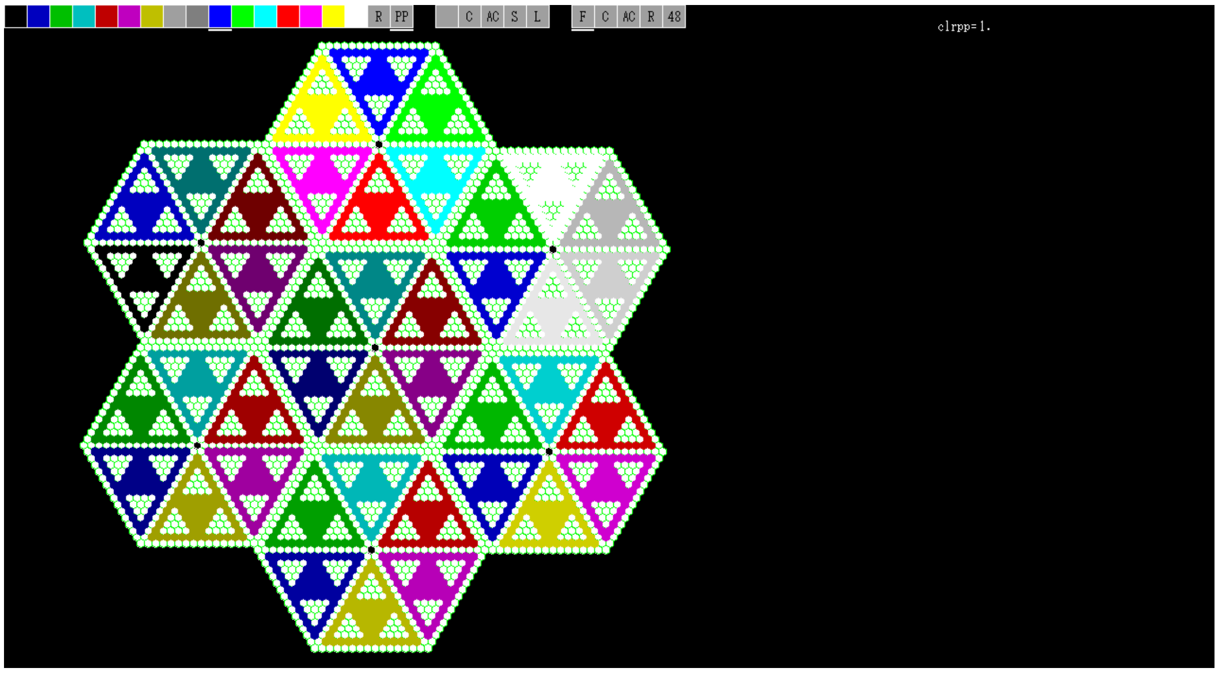

図 27

8. 配列

手作業で図形を規則的に作る場合は配列を二つ用います。図 28 に表示用配列と塗りつぶし支援用配列を示します。図中の数字はカラーコードです。

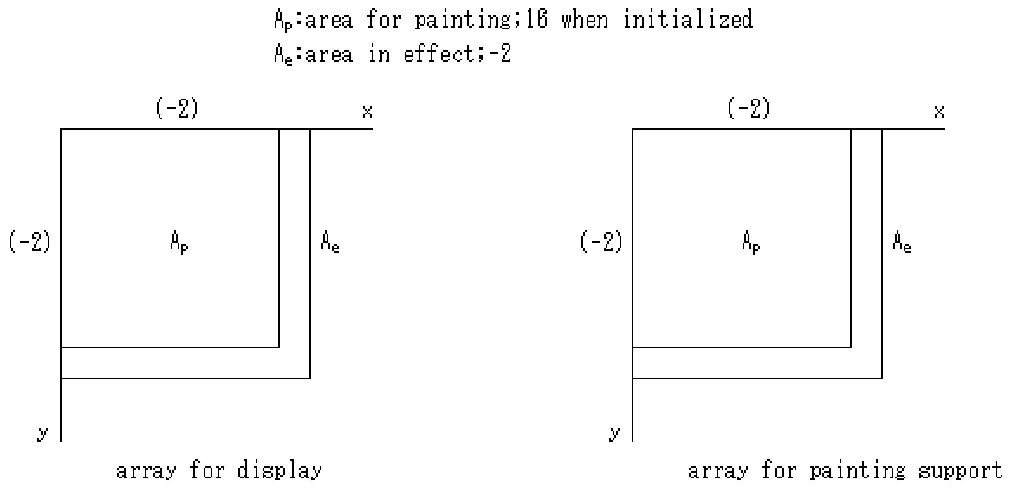

図 28

確保領域は塗りつぶし用領域より大き目に取ります。

第六回の図 10 の穴 A〜F、図 13 の穴 A〜D、第八回の図 14 の右の図形のカット部分も-2 に初期化します。(-2) は配列の添え字が負である場合は-2 を返すことを意味します。

塗りつぶし支援用配列は塗りつぶし検知法で種図形からコピー図形を作る場合に用います。塗りつぶし支援用配列における種図形のカラーコードは-1 です。

9. 適用

平行移動法は直交座標だけで使います。塗番号法は一点のコピーだけで使います。塗りつぶし検知法は全ての図形で使うことができます。これらの無限平面への適用の可能性を以下の表で示します。括弧内は色違いの塗りつぶしに関する適用の可能性です。

表 1　無限平面 (正方形ピクセル)

	一点	領域 A	領域 B	領域 C
平行移動法	○(○)	○(○)	○(○)	○(○)
塗番号法	○(○)	○(×)	○(×)	○(×)
塗りつぶし検知法	○(○)	○(○)	○(○)	○(○)

表 2　無限平面 (正六角形ピクセル)

	一点	領域 A	領域 B	領域 C
平行移動法	×	×	×	×
塗番号法	○(○)	○(△)	○(△)	○(△)
塗りつぶし検知法	○(○)	○(○)	○(○)	○(○)

領域 A、B、C は以下のようになっています。

・領域 A：C+S, C+0, 0+S
・領域 B：S′
・領域 C：C′

塗番号法では領域 A、B、C の色違いの塗りつぶしにおいてコピー図形が乱れます。図 29、30 はその例です。そのため塗番号法は主に塗りつぶし検知法におけるシードポイントのコピーに用います。

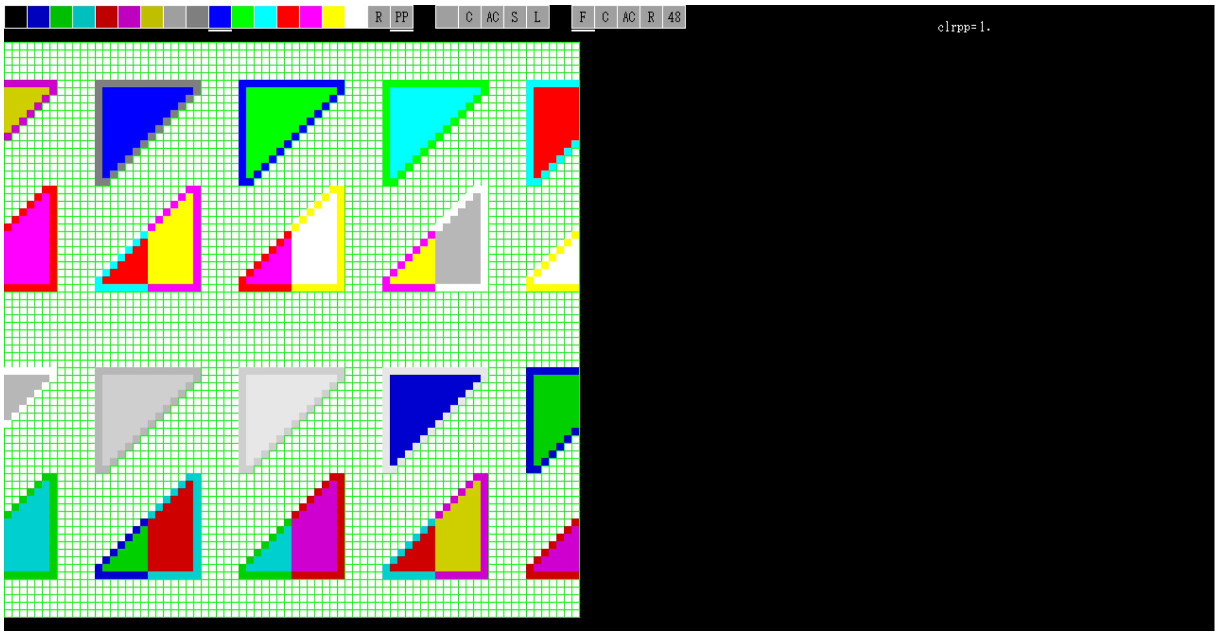

図 29

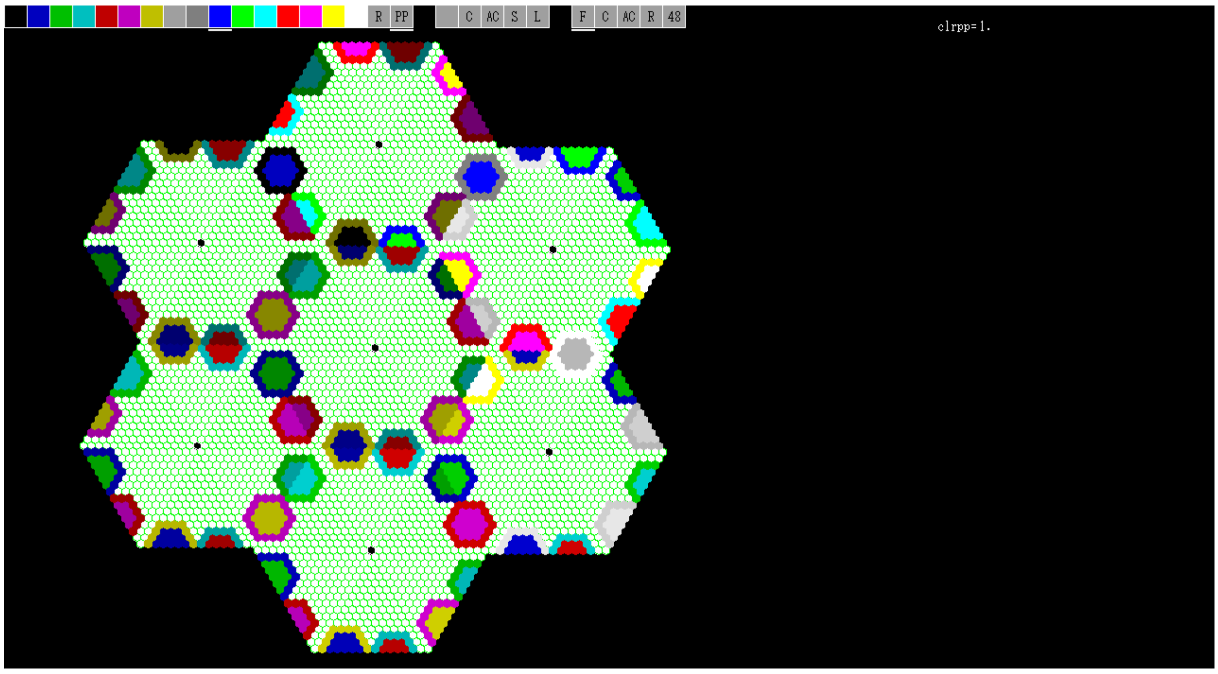

図 30

図多面体への適用の可能性を以下の表で示します。

表 3　多面体

	一点	領域 A	領域 B	領域 C
平行移動法	×	×	×	×
塗番号法	○(○)	○( )	○( )	○( )
塗りつぶし検知法	○(○)	○(○)	○(○)	○(○)

10. 過去分の修正

　第 6 回の void set_vals(int j) における文を修正して下さい。

・誤：

```
if(pixel[X][Y]==0) getpixel_(X_,Y_,X,Y);
    enX[j]=X;enY[j]=Y;enX_[j]=X_;enY_[j]=Y_;
```

・正：

```
if(Y<0 || X<0){ getpixel_(X_,Y_,X,Y);
    enX[j]=X;enY[j]=Y;enX_[j]=X_;enY_[j]=Y_;
}
```

一部のコンパイラでは pixel[0][-1]==0 となってしまいます。
以下をプロトタイプ宣言に加えて下さい。

```
/* 第７回 */
void pp_(int,int,int);

/* 第８回 */
void setstccolor(int);
void paint(char,int,int,int,int,int);
void C_and_S(char,int,int);
void arrayreset_(void);
void search(int,int,int);
void setcsrcolor(int);
void imm_check(void);
void imm_pause(void);
void imm_close(void);
void breaks(char);
#if WX==0
void WM_func_CHAR(WPARAM);
void WM_funcIME_CHAR(WPARAM);
void WM_funcIME_STARTCOMPOSITION(void);
void WM_funcIME_COMPOSITION(LPARAM);
void WM_funcIME_ENDCOMPOSITION(void);
#else
void WM_func_CHAR(unsigned char);
void WM_funcIME_CHAR(unsigned char *);
#endif
void csr(void);
void left_keydowns_dialog(void);
void backspace_dialog(void);
void text_end_dialog(void);
void page_down_dialog(void);
void page_up_dialog(void);
void trim_dialog(void);
void restore_dialog(void);
void clear_dialog(char);
void BitBlt_dialog(char);
void tailcheck_dialog(void);
void within_linemax_dialog(void);
void csr_row_home_dialog(void);
void csr_row_end_dialog(void);
void csr_right_dialog(void);
void fsave(void);
void fload(void);
void restore_edge(void);

/* 第9回 */
void setstccolor(int);
void paint(char,int,int,int,int,int);
void C_and_S(char,int,int);
void arrayreset_(void);
void search_bdr(int,int);
void search_i(int,int,int);
void setcsrcolor(int);
void imm_check(void);
void imm_pause(void);
void imm_close(void);
void breaks(char);
#if WX==0
void WM_func_CHAR(WPARAM);
void WM_funcIME_CHAR(WPARAM);
void WM_funcIME_STARTCOMPOSITION(void);
void WM_funcIME_COMPOSITION(LPARAM);
void WM_funcIME_ENDCOMPOSITION(void);
#else
void WM_func_CHAR(unsigned char);
void WM_funcIME_CHAR(unsigned char *);
#endif
void csr(void);
void left_keydowns_dialog(void);
void backspace_dialog(void);
void text_end_dialog(void);
void page_down_dialog(void);
void page_up_dialog(void);
void trim_dialog(void);
void restore_dialog(void);
void clear_dialog(char);
void BitBlt_dialog(char);
void tailcheck_dialog(void);
void within_linemax_dialog(void);
void csr_row_home_dialog(void);
void csr_row_end_dialog(void);
void csr_right_dialog(void);
void fsave(char *);
void fload(char *);
void restore_edge(void);
```

11. 具体例

プログラムを実行すると塗番号配列に座標を記憶する塗りつぶしが始まります。これが終わると塗りつぶし用領域が初期化されます。以下は塗りつぶしの手順です。

・F と PP を左クリックして下さい。
・左ボタンを押下しながらドラッグして 8 連結の閉曲線を作って下さい。
・閉曲線の内部のピクセルを Ctrl キーと Shit キーを押下しながら右クリックして下さい。C+S が得られます。
・パレットの緑を左クリックして下さい。
・塗りつぶされた閉曲線の内部の青いピクセルを右クリックして下さい。S′ が得られます。
・F と左の AC と F を順に左クリックして下さい。
・左ボタンを押下しながらドラッグして 8 連結の閉曲線を作って下さい。
・閉曲線の内部のピクセルを Ctrl キーを押下しながら右クリックして下さい。C+0 が得られます。
・F と左の AC と F を順に左クリックして下さい。
・左ボタンを押下しながらドラッグして 8 連結の閉曲線を作って下さい。
・閉曲線の内部のピクセルを Shift キーを押下しながら右クリックして下さい。0+S が得られます。
・F と左の AC と F を順に左クリックして下さい。
・左ボタンを押下しながらドラッグして 8 連結の領域を作って下さい。
・この領域のピクセルを Ctrl キーを押下しながら右クリックして下さい。C′ が得られます。

F の下にマーカーがある場合の操作です。

・左クリック：1 ピクセルの追加
・右の C：1 ピクセルの削除
・右の AC：全ピクセルの削除

塗りつぶしが終わった後の操作です。

・右の R：塗りつぶし前の状態の復元

塗りつぶし前の状態は tmp.bin にセーブされています。tmp.bin からのロードを行って塗りつぶし前の状態を復元します
F の下にマーカーがない場合の操作です。
・左クリック：1 ピクセルの追加
・左の C：1 ピクセルの削除
・左の AC：全ピクセルの削除
・S：ファイルへのセーブ
・L：ファイルからのロード

以下は記号定数です。

・`#define HEX_or_SQR`：ピクセルの選択 (0:正六角形ピクセル、1:正方形ピクセル)
・`#define Lhex`：D_1 の辺の正六角形ピクセル数
・`#define Lsqr`：(シードポイント間の正方形ピクセル数)/2
・`#define PIXSIZE`：ピクセルの大きさ
・`#define XRESO`：ウィンドウの横の大きさ

- `#define YRESO`：ウィンドウの縦の大きさ
- `#define X0`：グラフィクスの x 方向の位置
- `#define PHASE`：位相 (0:平行、1:反平行、2:斜め反平行)

****

List 1:cag_10.c

```
/* u2.7                                                                    */
/* 2019 Morio Kikuchi                                                      */

#define WX /*0*//*1*/0              /* 0:Windows, 1:Xlib */

#if WX==0
#include <windows.h>
#else
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xlocale.h>
#include <X11/cursorfont.h>
#include <X11/keysym.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define HEX_or_SQR /*0*//*1*/1
#define YOUR_ART /*0*//*1*/1
#if HEX_or_SQR==0
#define ASiZE (7*(3*Lhex*Lhex-3*Lhex+1))
#define RCMAX ((ASiZE-7)/CPMAX_)
#else
#define ASiZE (RESO*RESO)
#define RCMAX (ASiZE/CPMAX_)
#endif

#define VGACOLORS 50
#define dbl double
#define UdX 7
#define UdY 14
#define UDX (UdX+0)
#define UDY (UdY+2)
#if WX==0
#define POINT POINT
#define GKS GetKeyState
#define GKS_ GetKeyState
#else
#define FSIZE 14                    /* fontsize */
#define POINT XPoint
```

```
#define TRANS (65535./255)
#define XDSDY (asct+1)
#endif

#if HEX_or_SQR==0
#define CPMAX 42
#define CPGRP /*2*//*6*/6
#define CPMAX_ (CPGRP*7)
#define Lhex 16
#define PIXSIZE 8
#define RESO (6*Lhex-2)
#define XDP ((Lhex-1)*3+2)
#define YDP ((Lhex-1)*3+2)
#define X0 (200)
#define Y0 (0)

#define cpwidth 4
#define Lsqr 9
#define PIXSIZE_sqr 8
#define RESO_sqr (cpwidth*(2*Lsqr+1))
#else
#define CPMAX 16
#define CPMAX_ CPMAX
#define cpwidth 4
#define Lsqr 9
#define PIXSIZE 8
#define RESO (cpwidth*(2*Lsqr+1))
#define X0 (0)
#define Y0 (5)
#define PHASE /*0*//*1*//*2*/1
#endif

#define CPHALF CPMAX

#define XRESO 1280
#define YRESO 768
#define dyMAX 603
#define GRPH_0_MAX 4000
#define SQSZ 24
#define PPDY (SQSZ+10)
#define DX_ 141

#define ICEIL(a,b) (((a)+((b)-1))/(b))
#define Zx (XRESO*2)
#define Zy (YRESO*2)
#define Zxd2 (Zx/2)
```

```
#define Zyd2 (Zy/2)

#define CD 38                       /* COLUMN_DIALOG */
#define ASIZE (256+1)
#define ASIZEM 256
#define DI 2
#define DJ 0
#define DI_d (DI+2)                 /* d : dialog */
#define DJ_d (DJ+2)
#define DI_m 1
#define dummy_R 'R'

char refill,pauseflag,fieldflag,GRPH,EDGE,c_trans,d_trans,clrpp,Fill,searchflag;
char charcode,charflag,zeroFill,SPmode,_4_or_8;
int Zflag,X,Y,X_,Y_,dO[2],xg,yg,zg,xi,yi,zi;
int ca,c1,c2,c3,c4,c5,c7,c6,c8,color_old_g,wall=-2;
int nx[CPMAX],ny[CPMAX],nx_[CPMAX],ny_[CPMAX],std_x,std_y,last_x,last_y,
    nxp,nxm,nyp,nym,nxp_c,nxm_c,nyp_c,nym_c;
int enX[6+6],enY[6+6],enX_[6+6],enY_[6+6],enSN[6+6],jmp[6+6];
int ig,PIXSIZE_,idx,dy_hex,predraw,/*sn_,sn,*/bdrnum,bdrsnum;
int xt,yt,tmp0,tmp1,tmp2,tmp3,tmp4,putperiod;
long asize=ASiZE,d3;
long rcount[CPMAX],cnt;

char dialogflag,menuflag,filerflag,lumpflag_dialog,puts_mline_flag,overwriteflag,
     BitBltflag_,noclearflag,bitbltflag,cut,BitBltflag,cqflag,insorover,overwriteflag;
int icsr,jcsr,RTC=9,CSRDY=UDY;
long kmax_dialog,firstk_dialog;

char **pixel,p[ASIZE],p_dialog[ASIZE],p_restore[ASIZE],pixel_[XRESO][YRESO],
     cc[]="@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_";
long fp_mem[CPMAX],kmax;

char function,usflag;
unsigned char yorn;
int WB,WDX,WDY,Wdx,Wdy;
FILE *fp;

char dbflag,imm_restart_flag;
long dbsize;

#if WX==0
char immflag,imeendflag,compflag,dbcount;
#else
int argc_,asct;
char **argv_,appliname[]="CAG",fs1[ASIZE],stock_db[ASIZE];
```

```
long kmax_sdb;

char *fs2[5]={
              "-*-*-medium-r-normal--14*",       /* fn_set_0 (SFS) */
              "-*-*-medium-r-normal--20-*",       /* fn_set_1 (SFMM) */
              "-*-*-medium-r-normal--20-*",            /* fn_set_2 (SFM) */
              "-*-*-medium-r-normal--24-*",            /* fn_set_3 (SFL) */

              "none"
                                                  };  /* fontnames */
#endif

typedef struct {
short CPM;long RCM;short xt1,xt2,yt;} coordinates;
coordinates xy;
typedef struct {
int cx,cy,clr;} cag;
cag dsp[Zxd2][Zyd2];
cag bdr[ASiZE],bdrs[ASiZE];
typedef struct {
int x[CPMAX_][10],y[CPMAX_][10];} member;
member mbr[RCMAX];
typedef struct {
char p;dbl x,y,z;} swork;
swork work[Zx][Zy];
typedef struct {
int x,y;} sstack;
sstack stack[Zx*5]/*,stack_Rect[Zx*5]*/;
typedef struct {int xx,yy,xx_,yy_;} ss;
ss s;ss rtn[CPMAX][ASiZE];
typedef struct {
int back,fore;} bf;
bf bfset[]={{15,0},{0,15}};
#if WX==0
typedef struct {
unsigned char red,green,blue;} srgb;
srgb irgb[VGACOLORS];
#else
XColor irgb[VGACOLORS],c;
#endif

#if WX==0
HINSTANCE hinstance;
HWND hwnd;
HDC hdcdisplay,hdctmp1,hdctmp2,hdctmp3;
HBITMAP hbitmap1,hbitmap2,hbitmap3;
```

```
HPEN hpen;
HBRUSH hbrush;
HIMC himc,himc_;
COMPOSITIONFORM myime;
LOGFONT myimefont;
HFONT hfont;
POINT point;
RECT rect;
#else
Display *d;
int screen,depth;
Colormap cmap;
Window rw,w;
XSizeHints sh;
GC gcdisplay;
Pixmap pmap1,pmap2,pmap3;
XFontSet font_fs;
XFontStruct **info;
Cursor cursor;
XEvent event;
KeySym keysym,sym;
Visual *vis;
XImage *image;

unsigned long mask;
char **flist,**mlist,*def;
int mcount,fontnum;
XIM ime;
XIMStyle style;
XIC ic;
Status st;
#endif

void setstccolor(int);
void paint(char,int,int,int,int,int);
void C_and_S(int,int,int);
void arrayreset_(void);
void search_bdr(int,int);
void search_i(int,int,int);
void setcsrcolor(int);
void imm_check(void);
void imm_pause(void);
void imm_close(void);
void breaks(char);
#if WX==0
void WM_func_CHAR(WPARAM);
void WM_funcIME_CHAR(WPARAM);
void WM_funcIME_STARTCOMPOSITION(void);
void WM_funcIME_COMPOSITION(LPARAM);
void WM_funcIME_ENDCOMPOSITION(void);
#else
void WM_func_CHAR(unsigned char);
void WM_funcIME_CHAR(unsigned char *);
#endif
void csr(void);
void left_keydowns_dialog(void);
void backspace_dialog(void);
void text_end_dialog(void);
void page_down_dialog(void);
void page_up_dialog(void);
void trim_dialog(void);
void restore_dialog(void);
void clear_dialog(char);
void BitBlt_dialog(char);
void tailcheck_dialog(void);
void within_linemax_dialog(void);
void csr_row_home_dialog(void);
void csr_row_end_dialog(void);
void csr_right_dialog(void);
void fsave(char *);
void fload(char *);
void restore_edge(void);
void restore_magenta(int,int);
void search_i_cag_r(int,int,int,int);

void closegraph_(void),initpalette(void),BitBlt_full(void),setup(int),
     cleardevice_(char,int,int,int,int),rectangle_(char,int,int,int,int,int,int),
     delay_(long),beep(long),kbhit_(void),initgraph_return(void),
     use_subroop(void),keydowns_f2(void),bitblt(char,int,int,int,int,int,int),
     arrayreset(char,int),fwrite_mem(int),fread_mem(int),putpixel_(int,int,int),
     check_rcount(void),field(int),page_firstk_dialog(long),
     memcpy_(unsigned char*,long,unsigned char*,long,long),restore_in_PAINT(void);
char csr_left_dialog(void),gettype_dialog(long);
unsigned char subroop(void);
int initgraph_(void),setup_(void),fourfloor_fiveceil(dbl),random_(int),
    getpixel_(int,int),cag_r(int,int,int),

    deletion_dialog(void),scroll_down_dialog(void),scroll_up_dialog(void),
    insertion_dialog(unsigned char),ishead_dialog(long);
long ftell_mem(int);
dbl getangle(int,int);
```

```
#if WX==0
COLORREF PALETTE(int);
LRESULT CALLBACK wndproc_by_kbhit_(HWND,UINT,WPARAM,LPARAM);
int wndproc_filer(HWND,UINT,WPARAM,LPARAM);
#else
int wndproc_filer(void);
XIMStyle InputStyle(XIM);
XIC InputContext(XIM,XIMStyle,XFontSet,Window);
#endif


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

if(!YOUR_ART){
if(argc>1 && strcmp(argv[1],"0")==0){
GRPH=0;
if(argc==2) argc=1;else argc=2;
}
else GRPH=1;
}
else GRPH=1;
WB=1;
refill=1;
Zflag=1;
dO[0]=0;
dO[1]=0;
if(!YOUR_ART) {if(GRPH) predraw=1;else predraw=0;}
else          predraw=-1;

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

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

if(setup_()==1) return 1;
printf(" xt+1:%d\n",xt+1);
printf(" yt+1:%d\n",yt+1);
/*putpixel_noarray(RESO-1,RESO-1,8);*/
#if HEX_or_SQR==0
printf(" Lhex=%d ASiZE=%d RCMAX=%d\n",Lhex,ASiZE,RCMAX);
#else
printf(" Lsqr=%d RESO=%d RCMAX=%d\n",Lsqr,RESO,RCMAX);
#endif
arrayreset(0,0);
```

```
arrayreset_();

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

#if HEX_or_SQR==0
d3=Lhex+0;
#endif

while(1){
field(0);
/*putpixel_noarray(6*(RESO-1),0,8);*/
/*use_subroop();goto end;*/
if(YOUR_ART) predraw=1;
cag_r(-1,-1,-1);
if(refill==0) break;
check_rcount();
/*printf(" %d\n",d_trans);*/

#if !YOUR_ART
printf(" \n");
if(refill==0) break;
#else
delay_(2000);
arrayreset(1,-2);
c_trans=3;
field(-1);
printf(" \n");
refill=2;
xi=0;yi=0;                          /* for COPY=1 */
use_subroop();break;
#endif

if(GRPH){
beep(50);

delay_(6000);
if(pauseflag==1) {pauseflag=0;use_subroop();}
}/**if(GRPH)**/
if(refill==0) break;
arrayreset(2,0);
}/**while(1)**/

end:
closegraph_();
```

```
return 0;
}/** main **/


#if WX==0
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((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);

printf(" SAVE\n");
}
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 **/
#else
void ls_image(char flag,char *file,int x,int y,int dx,int dy)
{
unsigned long xsize,ysize,size;
unsigned long unitbytes,width,height,bits_per_pixel,bytes_per_line;
unsigned long i,j,k,k_,knew,oddbytes,dksum;
int c0,c1,c2;
unsigned long long heightdiv2,dbx;
unsigned char *buf_,*buf,*bf,*swap;
FILE *fpo,*fpi;

typedef struct {
unsigned char bfType[2];
unsigned long bfSize;
unsigned short bfReserved1;
unsigned short bfOffBits;
unsigned long bfReserved2;
} bfhset;
bfhset bfh;

typedef struct {
unsigned long biSize;
unsigned long biWidth;
unsigned long biHeight;
unsigned short biPlanes;
unsigned short biBitCount;
unsigned long biCompression;
unsigned long biSizeImage;
unsigned long biXPelsPerMeter;
unsigned long biYPelsPerMeter;
unsigned long biClrUsed;
unsigned long biClrImportant;
} bihset;
bihset bih;

if(flag<=3){                        /* save */
if(depth==16)      {unitbytes=2;printf(" 16bpp\n");}
```

```
else if(depth==24) {unitbytes=4;printf(" 24bpp\n");}
else               {printf("Depth unsuitable.\n");return;}

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

if(flag<=1)
/*image=XGetImage(d,pmap2,x,y,dx,dy,AllPlanes,ZPixmap)*/;
else if(flag==2)
image=XGetImage(d,pmap1,x,y,dx,dy,AllPlanes,ZPixmap);
else
image=XGetImage(d,rw,x,y,dx,dy,AllPlanes,ZPixmap);

width=image->width;
height=image->height;

oddbytes=(XRESO*3)%4;
if(oddbytes==0){
buf=(unsigned char *)malloc(1L*XRESO*3*YRESO);
swap=(unsigned char *)malloc(XRESO*3);
}
else{
buf=(unsigned char *)malloc(1L*(XRESO*3+(4-oddbytes))*YRESO);
swap=(unsigned char *)malloc(XRESO*3+(4-oddbytes));
}

if((oddbytes=(width*3)%4)==0){
size=width*height;
for(i=0;i<size;i++){
if(unitbytes==4){                    /* 4:24bpp, 2:16bpp */
buf[i*3+0]=image->data[i*4+0];
buf[i*3+1]=image->data[i*4+1];
buf[i*3+2]=image->data[i*4+2];
}
else{
c0=image->data[i*2+1];
c1=image->data[i*2+0];
                                     /* red, green, blue */
buf[i*3+2]=((c0 >> 3) & 31)*255/31;
buf[i*3+1]=(((c0 & 0x07) << 2) | ((c1 >> 6) & 0x03))*255/31;
buf[i*3+0]=(c1 & 31)*255/31;
}
}

bytes_per_line=width*3;
}/**if(oddbytes)**/
else{
```

```
k=0;k_=0;dksum=0;
for(j=0;j<height;j++){
for(i=0;i<width;i++){
if(unitbytes==4){
buf[k*3+0+dksum]=image->data[k_*4+0];
buf[k*3+1+dksum]=image->data[k_*4+1];
buf[k*3+2+dksum]=image->data[k_*4+2];
}
else{
c0=image->data[k_*2+1];
c1=image->data[k_*2+0];
                                    /* red, green, blue */
buf[k*3+2+dksum]=((c0 >> 3) & 31)*255/31;
buf[k*3+1+dksum]=(((c0 & 0x07) << 2) | ((c1 >> 6) & 0x03))*255/31;
buf[k*3+0+dksum]=(c1 & 31)*255/31;
}

k++;k_++;
}

if(unitbytes==2) k_+=(width*3)%2;   /* 16bpp */

knew=k*3+0+dksum;
for(i=0;i<4-oddbytes;i++){
buf[knew]=0;

knew++;
}

dksum+=(4-oddbytes);
}/**for(j)**/

bytes_per_line=width*3+(4-oddbytes);
}/**else(oddbytes)*/

/*printf(" size=%ld\n",bytes_per_line*height);
printf(" %d %d %d\n",width,height,width*height*3);*/

/*99*/
strcpy(bfh.bfType,"BM");
/*bfh.bfSize=bytes_per_line*height/65536;*/
bfh.bfSize=54+bytes_per_line*height;
bfh.bfReserved1=0;
bfh.bfOffBits=54;
bfh.bfReserved2=0;
```

```
bih.biSize=40;
bih.biWidth=width;
bih.biHeight=height;
bih.biPlanes=1;
bih.biBitCount=8*3;                  /* 24bpp */
bih.biCompression=0;
bih.biSizeImage=bytes_per_line*height;
bih.biXPelsPerMeter=2925;
bih.biYPelsPerMeter=2925;
bih.biClrUsed=0;
bih.biClrImportant=0;

size=bih.biSizeImage;
heightdiv2=height/2;
dbx=bytes_per_line;
for(i=0;i<heightdiv2;i++){
memmove(swap,&buf[i*dbx],dbx);
memmove(&buf[i*dbx],&buf[size-(i+1)*dbx],dbx);
memmove(&buf[size-(i+1)*dbx],swap,dbx);
}

fwrite(&bfh,14,1,fpo);
fwrite(&bih,40,1,fpo);
size=bih.biSizeImage;
fwrite(buf,size,1,fpo);

fclose(fpo);
free(buf);
free(swap);

printf(" SAVE\n");
}
else{                                /* load */
if(depth==16)      {printf("16bpp\n");}
else if(depth==24) {printf("24bpp\n");}
else               {printf("Depth unsuitable.\n");return;}

if((fpi=fopen(file,"rb"))==NULL) {printf("Can't open the file.\n");return;}

fread(&bfh,14,1,fpi);
if(strncmp(bfh.bfType,"BM",2)!=0) {fclose(fpi);printf("Not BM.\n");return;}
fread(&bih,40,1,fpi);

fseek(fpi,bfh.bfOffBits,0);
size=bih.biSizeImage;
buf_=(unsigned char *)malloc(size);
```

```
fread(buf_,size,1,fpi);

fclose(fpi);

width=bih.biWidth;
height=bih.biHeight;
bits_per_pixel=bih.biBitCount;
bytes_per_line=bih.biSizeImage/bih.biHeight;

oddbytes=(width*3)%4;
if(oddbytes==0)
swap=(unsigned char *)malloc(width*3);
else
swap=(unsigned char *)malloc(width*3+(4-oddbytes));

size=bih.biSizeImage;
heightdiv2=height/2;
dbx=bytes_per_line;
for(i=0;i<heightdiv2;i++){
memmove(swap,&buf_[i*dbx],dbx);
memmove(&buf_[i*dbx],&buf_[size-(i+1)*dbx],dbx);
memmove(&buf_[size-(i+1)*dbx],swap,dbx);
}

buf=(unsigned char *)malloc(width*height*3);
bf=(unsigned char *)malloc(width*height*4);

if((oddbytes=(width*3)%4)==0){
size=width*height;
for(i=0;i<size;i++){
buf[i*3+0]=buf_[i*3+0];
buf[i*3+1]=buf_[i*3+1];
buf[i*3+2]=buf_[i*3+2];
}
}/**if(oddbytes)**/
else{
k=0;k_=0;dksum=0;
for(j=0;j<height;j++){
for(i=0;i<width;i++){
buf[k*3+0]=buf_[k_*3+0+dksum];
buf[k*3+1]=buf_[k_*3+1+dksum];
buf[k*3+2]=buf_[k_*3+2+dksum];

k++;k_++;
}
dksum+=(4-oddbytes);
}/**for(j)**/
}/**if(oddbytes)**/

if(depth==16){
size=width*height;
for(i=0;i<size;i++){
/*c0=buf[i*3+0]*31/255;
c1=buf[i*3+1]*31/255;
c2=buf[i*3+2]*31/255;

buf[i*3+0]=0;
buf[i*3+1]=(c0<<3) | ((c1>>2) & 0x07);
buf[i*3+2]=(((c1 & 0x03) << 6) | c2)+32;*/

c0=buf[i*3+2]*31/255;                    /* bmp */
c1=buf[i*3+1]*31/255;
c2=buf[i*3+0]*31/255;

buf[i*3+2]=0;
buf[i*3+1]=(c0<<3) | ((c1>>2) & 0x07);
buf[i*3+0]=(((c1 & 0x03) << 6) | c2)+32;
}
}/**if(depth)**/

size=width*height;
for(i=0;i<size;i++){
bf[i*4+0]=buf[i*3+0];
bf[i*4+1]=buf[i*3+1];
bf[i*4+2]=buf[i*3+2];
}

vis=DefaultVisual(d,screen);

image=XCreateImage(d,vis,depth,ZPixmap,0,bf,width,height,32,width*4);

image->byte_order=LSBFirst;
image->bitmap_bit_order=LSBFirst;
image->bits_per_pixel=8*4;

/*XPutImage(d,pmap2,gcdisplay,image,0,0,x,y,width,height);*/

free(buf_);
free(buf);
free(bf);
free(swap);
```

```
}
}/** ls_image **/
#endif


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 **/


#if WX==0
void keydowns_f2(void)
{
int dy;

if(GKS(VK_ESCAPE)<0 || GKS(VK_PAUSE)<0) charflag=0;
else if(GKS('S')<0){
#if HEX_or_SQR==0
dy=Y0+5*9+(RESO_sqr-1+1/*1.7*/)*PIXSIZE_sqr+15+PPDY;
#else
dy=Y0+(RESO-1+1/*1.7*/)*PIXSIZE+15+PPDY;
#endif
ls_image(2,"ss.bmp",0,0,XRESO,YRESO/*dy*/);
beep(300);
```

```
}
}/** keydowns_f2 **/
#else
void keydowns_f2(void)
{
int dy;

if(GKS(XK_Escape)<0 || GKS(XK_Pause)<0) charflag=0;
else if(GKS('S')<0 || GKS('s')<0){
ls_image(2,"ss.bmp",0,0,XRESO,YRESO);
beep(300);
}
}/** keydowns_f2 **/
#endif


void restore_in_PAINT(void)
{
#if WX==0
ValidateRect(hwnd,NULL);
#endif

if(predraw>0) bitblt(1,0,0,XRESO,YRESO,0,0);
if(dialogflag) {BitBlt_dialog(2);csr();}
}/** restore_in_PAINT **/


#if WX==1
long max(long a,long b)
{
return (a>b)?a:b;
}/** max **/


long min(long a,long b)
{
return (a<b)?a:b;
}/** min **/
#endif


void setup(int flag)
{
if(flag==0){
if(YOUR_ART==0) EDGE=1;
else            EDGE=0;
```

```
Fill=-1;
}
#if WX==0
else if(flag==1){
WDX=GetSystemMetrics(SM_CXFIXEDFRAME);
WDY=GetSystemMetrics(SM_CYCAPTION)+GetSystemMetrics(SM_CYFIXEDFRAME);
}
else{
GetWindowRect(hwnd,&rect);
Wdx=rect.left;
Wdy=rect.top;
}
#endif
}/** setup **/


int get_dx_h(int nx,int ny)
{
int dx;

dx=ff_fc(X0+nx*1.0*PIXSIZE_-ny*0.5*PIXSIZE_);

/*if(nx<7*(RESO-1)) return (dx-200);
else              return (dx-300);*/
}/** get_dx_h **/


int get_dy_h(int nx,int ny)
{
int dy;

dy=ff_fc(Y0+ny*(sqrt(3)/2)*PIXSIZE_);

return dy;
}/** get_dy_h **/


int setup_(void)
{
int i,dy,PXSZ;

#if HEX_or_SQR==0
/*PXSZ=10;

dy=Y0+(yt+1)*PXSZ+10;
```

```
if(dy>dyMAX){
while(1){
PXSZ--;
dy=Y0+(yt+1)*PXSZ+10;
if(dy<=dyMAX) break;
}
}

if(PXSZ<4) PXSZ=4;*/
PIXSIZE_=/*ff_fc(PXSZ*sqrt(2))*/PIXSIZE;

dy=get_dy_h(1,1)-get_dy_h(0,0);
dy_hex=ff_fc(dy/3.);
#else
/*PXSZ=8;

reso:
while(1){
Lsqr=(RESO/cpwidth-1)/2;
RESO=cpwidth*(2*Lsqr+1);

if(Lsqr>=1) break;
else RESO++;
}

dy=YRESO/64;
while(1){
if(RESO*PXSZ+dy>YRESO){
if(Lsqr==1) PXSZ--;
else {RESO--;goto reso;}
}
else break;
}

if(PXSZ<4) PXSZ=4;*/
#endif

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

pixel=(/*unsigned */char **)malloc(sizeof(/*unsigned */char *)*((xt+1)+1));
if(pixel==NULL){
initgraph_return();return 1;}

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

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

i++;
if(i==(xt+1)+1) break;
}

return 0;
}/** setup_ **/


#if WX==0
void initsysfont(int type)
{
if(type==0){
hfont=CreateFont(UdY,UdX,0,0,
                 FW_NORMAL,FALSE,0,0,
                 DEFAULT_CHARSET,OUT_DEFAULT_PRECIS,
                 CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,
                 FIXED_PITCH | FF_ROMAN/*FF_MODERN*/,NULL);
}
else{
hfont=CreateFont(UdY,UdX,0,0,
                 FW_NORMAL,0,0,0,
                 DEFAULT_CHARSET,OUT_DEFAULT_PRECIS,
                 CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,
                 FIXED_PITCH | /*FF_ROMAN*/FF_MODERN,NULL);
}

SelectObject(hdcdisplay,hfont);
SelectObject(hdctmp1,hfont);
}/** initsysfont **/
#else
void initsysfont(int type)
{
if(type==0){                           /* small */
strcpy(fs1,fs2[0]);
strcat(fs1,"-*-*-*-*-*-*");            /* scalable */
```

```
font_fs=XCreateFontSet(d,fs1,&mlist,&mcount,&def);
XFontsOfFontSet(font_fs,&info,&flist);
asct=(*info)->ascent;
}
else if(type==-1){                  /* medium (math) */
strcpy(fs1,fs2[1]);
strcat(fs1,"-*-*-*-*-*-*");         /* scalable */

font_fs=XCreateFontSet(d,fs1,&mlist,&mcount,&def);
XFontsOfFontSet(font_fs,&info,&flist);
asct=(*info)->ascent;
}
else if(type==1){                   /* medium */
strcpy(fs1,fs2[2]);
strcat(fs1,"-*-*-*-*-*-*");         /* scalable */

font_fs=XCreateFontSet(d,fs1,&mlist,&mcount,&def);
XFontsOfFontSet(font_fs,&info,&flist);
asct=(*info)->ascent;
}
else{                               /* large */
strcpy(fs1,fs2[3]);
strcat(fs1,"-*-*-*-*-*-*");         /* scalable */

font_fs=XCreateFontSet(d,fs1,&mlist,&mcount,&def);

if(mcount>0)
font_fs=XCreateFontSet(d,"-*-*-medium-r-normal--14-*",&mlist,&mcount,&def);

XFontsOfFontSet(font_fs,&info,&flist);
asct=(*info)->ascent;
}
}/** initsysfont **/
#endif


int initgraph_(void)
{
#if WX==0
WNDCLASS wndclass;

setup(0);
hinstance=GetModuleHandle(NULL);

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,YRESO,
                  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);
hbitmap2=CreateCompatibleBitmap(hdcdisplay,XRESO,YRESO);
hbitmap3=CreateCompatibleBitmap(hdcdisplay,XRESO,UDY);

hdctmp1=CreateCompatibleDC(hdcdisplay);  /* text, dialog, menu */
hdctmp2=CreateCompatibleDC(hdcdisplay);  /* text, dialog, menu */
hdctmp3=CreateCompatibleDC(hdcdisplay);  /* cursor */

SelectObject(hdctmp1,hbitmap1);
SelectObject(hdctmp2,hbitmap2);
SelectObject(hdctmp3,hbitmap3);

SetBkMode(hdcdisplay,TRANSPARENT);
SetBkMode(hdctmp1,TRANSPARENT);
SetBkMode(hdctmp2,TRANSPARENT);
SetBkMode(hdctmp3,TRANSPARENT);

SetBkColor(hdcdisplay,PALETTE(bfset[WB].back));
SetBkColor(hdctmp1,PALETTE(bfset[WB].back));
SetBkColor(hdctmp2,PALETTE(bfset[WB].back));
SetBkColor(hdctmp3,PALETTE(bfset[WB].back));
#else
if((d=XOpenDisplay(""))==NULL) return 1;
screen=DefaultScreen(d);
cmap=DefaultColormap(d,screen);

setup(0);

rw=DefaultRootWindow(d);
w=XCreateSimpleWindow(d,rw,0,0,XRESO,YRESO,0,
                      irgb[bfset[WB].back].pixel,
                      irgb[bfset[WB].back].pixel);
sh.flags=PPosition | PSize;
sh.x=0;sh.y=0;
sh.width=XRESO;sh.height=YRESO;
XSetStandardProperties(d,w,appliname,appliname,None,argv_,argc_,&sh);

XSelectInput(d,w,KeyPressMask | ButtonPressMask | PointerMotionMask | ExposureMask);
XMoveWindow(d,w,0,0);
XMapWindow(d,w);
XFlush(d);

gcdisplay=XCreateGC(d,w,0,NULL);

depth=DefaultDepth(d,screen);
pmap1=XCreatePixmap(d,w,XRESO,YRESO,depth);  /* text, dialog, menu */
pmap2=XCreatePixmap(d,w,XRESO,YRESO,depth);  /* text, dialog, menu */
pmap3=XCreatePixmap(d,w,XRESO,UDY,depth);  /* cursor */

cursor=XCreateFontCursor(d,XC_arrow);
XDefineCursor(d,w,cursor);

XSetLineAttributes(d,gcdisplay,/*2*/1,LineSolid,CapButt,JoinMiter);
#endif

initsysfont(0);
#if WX==1
initIME();
#endif
initpalette();
setstccolor(bfset[WB].fore);
setcsrcolor(15);

setup(1);

return 0;
```

```
}/** initgraph_ **/


#if WX==0
void initgraph_return(void)
{
DeleteObject(hfont);
DeleteDC(hdctmp1);
DeleteDC(hdctmp2);
DeleteDC(hdctmp3);
DeleteObject(hbitmap1);
DeleteObject(hbitmap2);
DeleteObject(hbitmap3);

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

MessageBox(NULL,"Memory space is not left.","CAG",MB_OK);
}/** initgraph_return **/
#else
void initgraph_return(void)
{
XFreeFontSet(d,font_fs);
XFreeCursor(d,cursor);
XFreePixmap(d,pmap1);
XFreePixmap(d,pmap2);
XFreePixmap(d,pmap3);
XFreeGC(d,gcdisplay);

XFreeColormap(d,cmap);
XDestroyWindow(d,w);XFlush(d);
XCloseDisplay(d);

printf(" %s\n","Memory not enough");
}/** initgraph_return **/
#endif


void closegraph_(void)
{
int i;

i=0;
while(1){
```

```
free(pixel[i]);
i++;
if(i==(xt+1)+1) break;
}
free(pixel);

#if WX==0
DeleteObject(hfont);
DeleteObject(hbitmap1);
DeleteObject(hbitmap2);
DeleteObject(hbitmap3);
DeleteDC(hdctmp1);
DeleteDC(hdctmp2);
DeleteDC(hdctmp3);

/*EndPaint(hwnd,&paintstruct);*/
ReleaseDC(hwnd,hdcdisplay);
DestroyWindow(hwnd);
/*UnregisterClass("CAGCLASS",hinstance);*/
#else
XFreeFontSet(d,font_fs);
XFreeCursor(d,cursor);
XFreePixmap(d,pmap1);
XFreePixmap(d,pmap2);
XFreePixmap(d,pmap3);
XFreeGC(d,gcdisplay);

XFreeColormap(d,cmap);
XDestroyWindow(d,w);XFlush(d);
XCloseDisplay(d);
#endif
}/** 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=1;i<7;i++){                 /* 1 -> 6 */
if(irgb[9+(i-1)].red==255)
irgb[i].red=127+64;
if(irgb[9+(i-1)].green==255)
irgb[i].green=127+64;
if(irgb[9+(i-1)].blue==255)
irgb[i].blue=127+64;
}

for(i=7;i<9;i++){                 /* 7, 8 */
irgb[i].red=127+32*(8-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=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;
}

#if WX==1
for(i=0;i<VGACOLORS;i++){
irgb[i].red=fourfloor_fiveceil(irgb[i].red*TRANS);
irgb[i].green=fourfloor_fiveceil(irgb[i].green*TRANS);
irgb[i].blue=fourfloor_fiveceil(irgb[i].blue*TRANS);
}

for(i=0;i<VGACOLORS;i++)
XAllocColor(d,cmap,&irgb[i]);

XParseColor(d,cmap,"cyan",&c);
XAllocColor(d,cmap,&c);
#endif
}/** initpalette **/


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


#if WX==0
```

```
void bitblt(char flag,int x,int y,int xsize,int ysize,int x_,int y_)
{
int dy_=0;

if(bitbltflag==0){
if(flag==1)
BitBlt(hdcdisplay,x_,y_,xsize,ysize,
       hdctmp1,x,y,SRCCOPY);
else if(flag==2)
BitBlt(hdcdisplay,x_,y_,xsize,ysize,
       hdctmp2,x,y,SRCCOPY);
else if(flag==3)                    /* flag = 3 */ /* for csr() */
BitBlt(hdcdisplay,x_,y_,xsize,ysize,
       hdctmp2,/*x*/x_,/*y*/y_-dy_,SRCCOPY);
}/**if(bitbltflag)**/
else{
if(flag==1)
BitBlt(hdcdisplay,x_,y_,xsize,ysize,
       hdctmp1,x,y,SRCINVERT);
else if(flag==2)
BitBlt(hdcdisplay,x_,y_,xsize,ysize,
       hdctmp2,x,y,SRCINVERT);
else if(flag==-3)              /* flag = -3 */ /* for csr_to_1(), for csr_to_1_BL() */
BitBlt(hdctmp2,x_,y_-dy_,xsize,ysize,
       hdctmp3,x,y,SRCINVERT);         /* x, y = 0, 0 */
}/**else(bitbltflag)**/
}/** bitblt **/
#else
void bitblt(char flag,int x,int y,int xsize,int ysize,int x_,int y_)
{
if(bitbltflag==0){
if(flag==1){
/*printf(" %d %d %d %d %d %d\n",x,y,x_,y_,xsize,ysize);*/
XCopyArea(d,pmap1,w,gcdisplay,x,y,xsize,ysize,
                              x_,y_);
}
else if(flag==2)
XCopyArea(d,pmap2,w,gcdisplay,x,y,xsize,ysize,
                                  x_,y_);
else if(flag==3)                    /* flag = 3 */ /* for csr() */
XCopyArea(d,pmap2,w,gcdisplay,/*x*/x_,/*y*/y_,xsize,ysize,
                              x_,y_);
}/**if(bitbltflag)**/
else{
if(flag==1)
XCopyArea(d,pmap1,w,gcdisplay,x,y,xsize,ysize,
                                x_,y_);
else if(flag==2)
XCopyArea(d,pmap2,w,gcdisplay,x,y,xsize,ysize,
                                    x_,y_);
else if(flag==-3)               /* flag = -3 */ /* for csr_to_1(), for csr_to_1_BL() */
XCopyArea(d,pmap3,pmap2,gcdisplay,/*x*/0,/*y*/0,xsize,ysize,
                                    x_,y_);
}/**else(bitbltflag)**/

XFlush(d);
}/** bitblt **/
#endif



#if WX==0
void cleardevice_(char hdc,int x,int y,int xsize,int ysize)
{
if(hdc==0)
PatBlt(hdcdisplay,x,y,xsize,ysize,PALETTE(bfset[WB].back));
else if(hdc==1)
PatBlt(hdctmp1,x,y,xsize,ysize,PALETTE(bfset[WB].back));
else
PatBlt(hdctmp2,x,y,xsize,ysize,PALETTE(bfset[WB].back));
}/** cleardevice_ **/
#else
void cleardevice_(char hdc,int x,int y,int xsize,int ysize)
{
XSetForeground(d,gcdisplay,irgb[bfset[WB].back].pixel);

if(hdc==0)
XFillRectangle(d,w,gcdisplay,x,y,xsize,ysize);
else if(hdc==1)
XFillRectangle(d,pmap1,gcdisplay,x,y,xsize,ysize);
else
XFillRectangle(d,pmap2,gcdisplay,x,y,xsize,ysize);
}/** cleardevice_ **/
#endif



#if WX==0
void setstccolor(int color)
{
SetTextColor(hdcdisplay,PALETTE(color));
SetTextColor(hdctmp1,PALETTE(color));
SetTextColor(hdctmp2,PALETTE(color));
}/** setstccolor **/
```

```
#else
void setstccolor(int color)
{
XSetForeground(d,gcdisplay,irgb[color].pixel);
}/** setstccolor **/
#endif


#if WX==0
void stc(char hdc,int x,int y,unsigned char *str,int size)
{
if(hdc!=0)
TextOut(hdcdisplay,x,y,str,size);

if(hdc==0)
TextOut(hdcdisplay,x,y,str,size);
else if(hdc==1)
TextOut(hdctmp1,x,y,str,size);
else
TextOut(hdctmp2,x,y,str,size);
}/** stc **/
#else
void stc(char hdc,int x,int y,unsigned char *str,int size)
{
if(hdc!=0)
XmbDrawString(d,w,font_fs,gcdisplay,x,y+XDSDY,str,size);

if(hdc==0)
XmbDrawString(d,w,font_fs,gcdisplay,x,y+XDSDY,str,size);
else if(hdc==1)
XmbDrawString(d,pmap1,font_fs,gcdisplay,x,y+XDSDY,str,size);
else
XmbDrawString(d,pmap2,font_fs,gcdisplay,x,y+XDSDY,str,size);
}/** stc **/
#endif


#if WX==0
char gettype(char flag,unsigned char s1,long k,long kend)
{
char type;

/* code page:932(SJIS)-> */
if(s1>=0x20 && s1<=0x7e) type=0;                   /* word */
else if(s1>=0xa1 && s1<=0xdf) type=0;
else if(flag==1 && k==kend) type=0;
else if(s1==0x0a) type=0;
else if(s1==0x09) type=1;
else if((s1>0x00 && s1<0x20) || s1==0x7f) type=2;
else if(s1<=0x7f) type=-1;                          /* others */
else type=3;                                        /* Double Byte Character */
/* <-code page:932(SJIS) */
/*else if(s1>=0x81 && s1<=0xfc && (s1<=0x9f || s1>=0xe0) &&
        s2>=0x40 && s2<=0xfc && s2!=0x7f) type=3;*/  /* Double Byte Character */

return type;
}/** gettype **/
#else
char gettype(char flag,unsigned char s1,long k,long kend)
{
char type;

/* EUC-> */
if(s1>=0x20 && s1<=0x7e) type=0;                    /* word */
else if(flag==1 && k==kend) type=0;
else if(s1==0x0a) type=0;                           /* control code(LF) */
else if(s1==0x09) type=1;                           /* control code(HT) */
else if((s1>0x00 && s1<0x20) || s1==0x7f) type=2;  /* control code(others) */
else if(s1<=0x7f) type=-1;                          /* others */
else type=3;                                        /* Double Byte Character */
/* <-EUC */
/*else if(s1>=0x81 && s1<=0xfc && (s1<=0x9f || s1>=0xe0) &&
        s2>=0x40 && s2<=0xfc && s2!=0x7f) type=3;*/  /* Double Byte Character */

return type;
}/** gettype **/
#endif

int while_puts_show_(int xlast,int ylast)
{
char type;
int i,j,dx,dy;
long k;
unsigned char s[1];
unsigned char jis[2];

i=xlast;j=ylast;
k=0;

while(1){
s[0]=p[k];
type=gettype(1,s[0],k,kmax);
if(type<=2){
dx=(i+0)*UDX;dy=(j+0)*UDY;
stc(1,dx,dy,s,1);

if(k==kmax) break;

k++;

i++;
}/**if(type)**/
else if(type==3){
jis[0]=p[k];
jis[1]=p[k+1];

dx=(i+0)*UDX;dy=(j+0)*UDY;
stc(1,dx,dy,jis,2);

if(/*k==kmax-1 || */k==kmax) break;              /* ? */

k+=2;
}/**else if(type)**/
else{
}/**else(type)**/

}

return i;
}/** while_puts_show_ **/


void printf_(char *str,dbl val,long i,long j)
{
char pflag=1;
int dx,dy;
int columns,len1,len2;
unsigned char buf[11];

dx=i*UDX;dy=j*UDY;

if(pflag==1) columns=7;
else         columns=11;

/*itoa(val,buf,10);*/gcvt(val,columns-1,buf);
len1=strlen(str);
len2=strlen(buf);
```

```
cleardevice_(0,dx,dy,UDX*(len1+7),UDY);  /* instead of bitblt(pflag,...) */
if(pflag==1)
paint(pflag,dx,dy,UDX*(len1+7),UDY+0,bfset[WB].back);

/* string */
setstccolor(bfset[WB].fore);
/*stc(1,dx,dy,str,strlen(str));*/
strcpy(p,str);kmax=strlen(p)-1;
while_puts_show_(i,j);

/* value */
stc(pflag,dx+len1*UDX+2,dy,buf,len2);

if(/*pflag==1*/0)
bitblt(pflag,dx,dy,UDX*(len1+7),UDY,dx,dy);
}/** printf_ **/



#if WX==0
void paint(char hdc,int x,int y,int xsize,int ysize,int color)
{
hbrush=CreateSolidBrush(PALETTE(color));

if(hdc==0){
SelectObject(hdcdisplay,hbrush);
PatBlt(hdcdisplay,x,y,xsize,ysize,PATCOPY);
}
else if(hdc==1){
SelectObject(hdctmp1,hbrush);
PatBlt(hdctmp1,x,y,xsize,ysize,PATCOPY);
}
else{
SelectObject(hdctmp2,hbrush);
PatBlt(hdctmp2,x,y,xsize,ysize,PATCOPY);
}

DeleteObject(hbrush);
}/** paint **/
#else
void paint(char hdc,int x,int y,int xsize,int ysize,int color)
{
XSetForeground(d,gcdisplay,irgb[color].pixel);

if(hdc==0)
XFillRectangle(d,w,gcdisplay,x,y,xsize,ysize);
```

```
else if(hdc==1)
XFillRectangle(d,pmap1,gcdisplay,x,y,xsize,ysize);
else
XFillRectangle(d,pmap2,gcdisplay,x,y,xsize,ysize);
}/** paint **/
#endif


#if WX==0
COLORREF PALETTE(int color)
{
return RGB(irgb[color].red,irgb[color].green,irgb[color].blue);
}/** PALETTE **/
#endif


#if WX==0
void kbhit_(void)
{
MSG msg;

if(PeekMessage(&msg,NULL,0,0,PM_REMOVE)){
TranslateMessage(&msg);
DispatchMessage(&msg);
}
}/** kbhit_ */
#else
int GKS(KeySym XK)
{
if(keysym==XK) return -1;
else return 0;
}/** GKS **/


int GKS_(long ModkeyMask)
{
if((event.xkey.state & ModkeyMask)>0) return -1;
else return 0;
}/** GKS_ **/


void kbhit_(void)
{
if(XPending(d)){
XNextEvent(d,&event);
```

```
wndproc_filer();
}
}/** kbhit_ */
#endif



#if WX==0
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_ **/
#endif



#if WX==0
int wndproc_filer(HWND hwnd,UINT umsg,WPARAM wparam,LPARAM lparam)
{
char gotoflag,BDflag,old,old_;
int x[2],y[2],z[2],dlt=SQSZ,val,dx,dy,i,j;
static int pcolor_old=8,pcolor=9,procedure=0;
POINT cpos;

setup(2);

if(umsg==WM_KEYDOWN){
/************************** menu keydowns -> ****************************/
/************************** <- menu keydowns ****************************/

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

if(dialogflag>0){

imm_check();

if(immflag==2) immflag=0;
if(usflag==1) usflag=0;

if(compflag) return 1;

if(cqflag==2){
  BitBltflag_=2;
  goto end_dialog;}
if(cqflag==6){
  /*BitBltflag_=2;*/
  goto end_left_dialog;}

gotoflag=1;

if(GKS(VK_ESCAPE)<0 || GKS(VK_PAUSE)<0 || GKS(VK_F12)<0 || GKS(VK_F1)<0){  /* added */
  dialogflag=3;refill=0;BitBltflag_=2;}
else if(GKS(VK_RETURN)<0){
  trim_dialog();
  dialogflag=2;refill=0;BitBltflag_=2;}

else {gotoflag=0;}

if(gotoflag==1) goto end_dialog;

end_left_dialog:
left_keydowns_dialog();

end_dialog:
if(BitBltflag_==0) {BitBlt_dialog(2);csr();}
else if(BitBltflag_==1)             csr();
else{}

return 1;
}/**if(dialogflag)**/

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

if(function==2){
imm_pause();
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_CHAR){
  WM_func_CHAR(wparam);
return 1;
}/**else if(umsg)**/
```

```
else if(umsg==WM_IME_CHAR){
  WM_funcIME_CHAR(wparam);
return 1;
}/**else if(umsg)**/
else if(umsg==WM_IME_STARTCOMPOSITION){
  WM_funcIME_STARTCOMPOSITION();
}/**else if(umsg)**/
else if(umsg==WM_IME_COMPOSITION){
  WM_funcIME_COMPOSITION(lparam);
}/**else if(umsg)**/
else if(umsg==WM_IME_ENDCOMPOSITION){
  WM_funcIME_ENDCOMPOSITION();
}/**else if(umsg)**/
else if(umsg==WM_CLOSE){
imm_close();
breaks(0);

if(dialogflag>0){
dialogflag=3;refill=0;
}
else{
refill=0;if(GKS_(VK_SHIFT)<0) refill--;charflag=0;charcode=2;
}

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

return 1;
}/**else if(umsg)**/
else if(umsg==WM_LBUTTONDOWN){
if(YOUR_ART==0) return 1;
if(refill==1 || dialogflag>0) return 1;

GetCursorPos(&cpos);
dx=cpos.x-WDX-Wdx;dy=cpos.y-WDY-Wdy;
x[0]=dsp[dx][dy].cx;y[0]=dsp[dx][dy].cy;

if(x[0]<0 || y[0]<0){
if(Fill==-1){
if(procedure==0){
BDflag=getflag(dx,dy);
if(BDflag==3) return 1;
}
else{
```

```
rectangle_(0,19*dlt,dlt-PPDY+2,(19+2)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
procedure=0;
return 1;
}
}/**if(Fill)**/
else{
if(procedure==0){
BDflag=getflag(dx,dy);
if(BDflag==3) return 1;
}
else{
rectangle_(0,26*dlt,dlt-PPDY+2,(26+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
procedure=0;
return 1;
}
}/**else(Fill)**/
}/**if(x[0],y[0])**/

if(Fill==-1){
if(procedure==0){
BDflag=getflag(dx,dy);

if(BDflag==0 && xg==17 && clrpp==1){
rectangle_(0,xg*dlt,dlt-PPDY+2,(xg+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
}

if(BDflag==0){
if(xg==16){
restore_edge();
rectangle_(0,16*dlt,dlt-PPDY+2,(16+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
printf(" Restoration of edge\n");
}
else if(xg==17){
if(!clrpp) clrpp=1;else clrpp=0;
printf_("clrpp=",clrpp,DX_,2-1);
printf(" color PlusPlus\n");
}
else{
pcolor_old=pcolor;
pcolor=xg;
printf(" color=%d old=%d\n",xg,pcolor_old);
}
}/**if(BDflag)**/
else if(BDflag==1){
     if(xg==0) procedure=1;             /* ID_1st */
else if(xg==1) procedure=2;             /* Clear */
else if(xg==2){
field(-1);
rectangle_(0,21*dlt,dlt-PPDY+2,(21+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
printf(" All Clear\n");
}
else if(xg==3){
refill=1;
fsave("");
rectangle_(0,22*dlt,dlt-PPDY+2,(22+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
refill=2;
}
else if(xg==4){
refill=1;
fload("");
rectangle_(0,23*dlt,dlt-PPDY+2,(23+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
refill=2;
}

     if(xg==0)
printf(" change ID_1st\n");
else if(xg==1)
printf(" Clear\n");
}/**else if(BDflag)**/
else if(BDflag==2){
     if(xg==0){
Fill=0;bdrnum=0;
printf(" into Fill(Fill:%d)\n",Fill);
}
}/**else if(BDflag)**/
else{
search_i(x[0],y[0],pcolor);
}/**else(BDflag)**/
}/**if(procedure)**/
else if(procedure==1){
if(x[0]>=0 && y[0]>=0 && (val=pixel[x[0]][y[0]])>=0 && val<=15){
/*id_1st=id[x[0]][y[0]];*/              /* change 1st */
/*BitBlt_full();*/
/*printf_("id_1st=",id_1st,DX_,0);*/
/*printf(" new ID_1st=%d\n",id_1st);*/
}
else{                                 /* id[][]=-2, -1 */
printf(" unpointed\n");
}

rectangle_(0,19*dlt,dlt-PPDY+2,(19+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
procedure=0;
```

```
}/**else if(procedure)**/
else if(procedure==2){
/*if(x[0]<0 || y[0]<0) printf(" ?\n");*/
if(x[0]>=0 && y[0]>=0 && (val=pixel[x[0]][y[0]])>=0 && val<=15){
/*id[x[0]][y[0]]=-1;*/                  /* restore one */
old=EDGE;EDGE=1;
search_i(x[0],y[0],16);
EDGE=old;
/*BitBlt_full();*/
}
else{
printf(" unpointed\n");
}

rectangle_(0,20*dlt,dlt-PPDY+2,(20+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
procedure=0;
}/**else if(procedure)**/
}/**if(Fill)****************************************************************/
else{
if(procedure==0){
BDflag=getflag(dx,dy);

if(BDflag==0 && xg==17 && clrpp==1){
rectangle_(0,xg*dlt,dlt-PPDY+2,(xg+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
}

if(BDflag==0){
if(xg==16){
restore_edge();
rectangle_(0,16*dlt,dlt-PPDY+2,(16+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
printf(" Restoration of edge\n");
}
else if(xg==17){
if(!clrpp) clrpp=1;else clrpp=0;
printf_("clrpp=",clrpp,DX_,2-1);
printf(" color PlusPlus\n");
}
else{
pcolor_old=pcolor;
pcolor=xg;
printf(" color=%d old=%d\n",xg,pcolor_old);
}
}/**if(BDflag)**/
else if(BDflag==1){
}/**else if(BDflag)**/
else if(BDflag==2){
     if(xg==0){
Fill=-1;
C_and_S(-1,-1,-1);
printf(" out of Fill(Fill:%d)\n",Fill);
}
else if(xg==1) procedure=1;         /* C */
else if(xg==2){                     /* AC */
Fill=0;
C_and_S(-1,-1,-1);
bdrnum=0;
rectangle_(0,(25+2)*dlt,dlt-PPDY+2,(25+3)*dlt,dlt-PPDY+3,bfset[WB].back,0);/* erase */
}
else if(xg==3){                     /* R */
/*refill=1;*/
fload("tmp.bin");
rectangle_(0,28*dlt,dlt-PPDY+2,(28+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
/*refill=2;*/
}
else if(xg==4){                     /* 48 */
#if HEX_or_SQR==1
if(!_4_or_8) _4_or_8=1;else _4_or_8=0;
printf_("_4_or_8=",_4_or_8,DX_,1-1);
printf(" 4_or_8\n");
rectangle_(0,29*dlt,dlt-PPDY+2,(29+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
#endif
}
}/**else if(BDflag)**/
else{
if(/*GKS_(VK_CONTROL)*/x[0]>=0 && y[0]>=0){
if(Fill==0){
search_bdr(x[0],y[0]);
}
else{
if((i=bdrcheck(x[0],y[0]))<0) search_bdr(x[0],y[0]);
}
}/**if(GKS_(VK_CONTROL))**/
}/**else(BDflag)**/
}/**if(procedure)**/
else if(procedure==1){
restore_magenta(x[0],y[0]);

rectangle_(0,(25+1)*dlt,dlt-PPDY+2,(25+2)*dlt,dlt-PPDY+3,bfset[WB].back,0);/* erase */
procedure=0;
}/**else if(procedure)**/
}/**else(Fill)***********************************************************/
```

```
return 1;
}/**else if(umsg)**/
else if(umsg==WM_RBUTTONDOWN){      /* R_BUTTON */
if(YOUR_ART==0) return 1;
if(refill==1 || dialogflag>0) return 1;
if(Fill<0) return 1;

GetCursorPos(&cpos);
dx=cpos.x-WDX-Wdx;dy=cpos.y-WDY-Wdy;
x[0]=dsp[dx][dy].cx;y[0]=dsp[dx][dy].cy;

if(x[0]<0 || y[0]<0) return 1;

     if(Fill>0 && GKS_(VK_CONTROL)<0 && GKS_(VK_SHIFT)>=0) val=0;   /* C+0 */
else if(Fill>0 && GKS_(VK_CONTROL)>=0 && GKS_(VK_SHIFT)<0) val=1;   /* 0+S */
else if(Fill>0 && GKS_(VK_CONTROL)>=0 && GKS_(VK_SHIFT)>=0) val=2;  /* C+S */
else if(Fill>0 && GKS_(VK_CONTROL)<0 && GKS_(VK_SHIFT)<0) val=3;    /* C+S */
else if(Fill==0) val=4;                                             /* 0+S */
Fill=0;
xg=x[0];yg=y[0];
/*999*/
     if(val==0) {val=0;C_and_S(val,-1,pcolor);}  /* S, C */
else if(val==1) {val=1;C_and_S(val,pcolor,-1);}
else if(val==2) {val=2;C_and_S(val,pcolor,pcolor);}
else if(val==3) {val=2;C_and_S(val,pcolor,pcolor_old);}
else if(val==4) {val=3;C_and_S(val,pcolor,-1);}

bdrnum=0;
bdrsnum=0;SPmode=0;
/*xi=0;yi=0;*/                            /* for COPY==1 */
printf(" done\n");

return 1;
}/**else if(umsg)**/
else if(umsg==WM_MOUSEMOVE){
if(YOUR_ART==0) return 1;
if(refill==1 || dialogflag>0/* || Fill==-1*/) return 1;
if(!(wparam & MK_LBUTTON)) return 1;

GetCursorPos(&cpos);
dx=cpos.x-WDX-Wdx;dy=cpos.y-WDY-Wdy;
x[0]=dsp[dx][dy].cx;y[0]=dsp[dx][dy].cy;

if(x[0]<0 || y[0]<0) return 1;

if(Fill<0){
```

```
search_i(x[0],y[0],pcolor);
}/**if(Fill)**/
else{
if(Fill==0){
search_bdr(x[0],y[0]);
}
else{
if((i=bdrcheck(x[0],y[0]))<0) search_bdr(x[0],y[0]);
}
}/**else(Fill)**/

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

return 0;
}/** wndproc_filer **/
#else
int wndproc_filer(void)
{
char gotoflag,buf[10],BDflag,old,old_;
int length,x[2],y[2],z[2],dlt=SQSZ,val,dx,dy,i,j;
static int pcolor_old=8,pcolor=9,procedure=0;
unsigned char buf_Xmb[ASIZE];
Window root,child;
int rx,ry,wx,wy;
unsigned int mask;

entrance:
/*length=*/XLookupString((XKeyEvent *)&event,buf,10,&keysym,NULL);
/*buf[length]='\0';*/

length=XmbLookupString(ic,(XKeyEvent *)&event,buf_Xmb,ASIZE,&sym,&st);
if(st==XLookupBoth || st==XLookupChars){}
else length=0;
if(length>0) buf_Xmb[length]='\0';

if(dialogflag>0) imm_restart();

if(event.type==KeyPress){
/************************** menu keydowns -> ****************************/
/************************** <- menu keydowns ****************************/

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

if(dialogflag>0){
```

```
if(usflag==1) usflag=0;

if(cqflag==2){
  BitBltflag_=2;
  goto end_dialog;}
if(cqflag==6){
  /*BitBltflag_=2;*/
  goto end_left_dialog;}

gotoflag=1;

if(GKS(XK_Escape)<0 || GKS(XK_Pause)<0 || GKS(XK_F12)<0 || GKS(XK_F1)<0){
  imm_pause();
  dialogflag=3;refill=0;BitBltflag_=2;}
else if(GKS(XK_Return)<0){
  imm_pause();
  trim_dialog();
  dialogflag=2;refill=0;BitBltflag_=2;}

else {gotoflag=0;}

if(gotoflag==1) goto end_dialog;

end_left_dialog:
left_keydowns_dialog();

end_dialog:
if(length>0){
if(length>1)
WM_funcIME_CHAR(buf_Xmb);                  /* double byte */
else if(length==1 && GKS_(ControlMask)>=0 && GKS_(Mod1Mask)>=0 &&
        buf_Xmb[0]>=0x20 && buf_Xmb[0]<=0x7e)
WM_func_CHAR(buf_Xmb[0]);           /* only ASCII CODE */
else
WM_func_CHAR(0);
}

if(BitBltflag_==0) {BitBlt_dialog(2);csr();}
else if(BitBltflag_==1)             csr();
else{}

return 1;
}/**if(dialogflag)**/

/************************** <- dialog keydowns ****************************/
```

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

if(usflag==1) usflag=0;

     if(GKS(XK_Escape)<0 || GKS(XK_Pause)<0) refill=0;
else if(GKS(XK_Shift_L)<0 || GKS(XK_Shift_R)<0) pauseflag=1;

return 1;
}/**if(event.type)**/
else if(event.type==Expose){
restore_in_PAINT();

return 1;
}/**else if(event.type)**/
else if(event.type==ButtonPress){
if(YOUR_ART==0) return 1;
if(refill==1 || dialogflag>0) return 1;

dx=event.xbutton.x;dy=event.xbutton.y;
x[0]=dsp[dx][dy].cx;y[0]=dsp[dx][dy].cy;

if(event.xbutton.button==1){        /* L_BUTTON */
if(x[0]<0 || y[0]<0){
if(Fill==-1){
if(procedure==0){
BDflag=getflag(dx,dy);
if(BDflag==3) return 1;
}
else{
rectangle_(0,19*dlt,dlt-PPDY+2,(19+2)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
procedure=0;
return 1;
}
}/**if(Fill)**/
else{
if(procedure==0){
BDflag=getflag(dx,dy);
if(BDflag==3) return 1;
}
else{
rectangle_(0,26*dlt,dlt-PPDY+2,(26+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
procedure=0;
return 1;
```

```
}
}/**else(Fill)**/
}/**if(x[0],y[0])**/

if(Fill==-1){
if(procedure==0){
BDflag=getflag(dx,dy);

if(BDflag==0 && xg==17 && clrpp==1){
rectangle_(0,xg*dlt,dlt-PPDY+2,(xg+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
}

if(BDflag==0){
if(xg==16){
restore_edge();
rectangle_(0,16*dlt,dlt-PPDY+2,(16+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
printf(" Restoration of edge\n");
}
else if(xg==17){
if(!clrpp) clrpp=1;else clrpp=0;
printf_("clrpp=",clrpp,DX_,2-1);
printf(" color PlusPlus\n");
}
else{
pcolor_old=pcolor;
pcolor=xg;
printf(" color=%d old=%d\n",xg,pcolor_old);
}
}/**if(BDflag)**/
else if(BDflag==1){
     if(xg==0) procedure=1;            /* ID_1st */
else if(xg==1) procedure=2;            /* Clear */
else if(xg==2){
field(-1);
rectangle_(0,21*dlt,dlt-PPDY+2,(21+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
printf(" All Clear\n");
}
else if(xg==3){
refill=1;
fsave("");
rectangle_(0,22*dlt,dlt-PPDY+2,(22+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
refill=2;
}
else if(xg==4){
refill=1;
fload("");
rectangle_(0,23*dlt,dlt-PPDY+2,(23+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
refill=2;
}

     if(xg==0)
printf(" change ID_1st\n");
else if(xg==1)
printf(" Clear\n");
}/**else if(BDflag)**/
else if(BDflag==2){
     if(xg==0){
Fill=0;bdrnum=0;
printf(" into Fill(Fill:%d)\n",Fill);
}
}/**else if(BDflag)**/
else{
search_i(x[0],y[0],pcolor);
}/**else(BDflag)**/
}/**if(procedure)**/
else if(procedure==1){
if(x[0]>=0 && y[0]>=0 && (val=pixel[x[0]][y[0]])>=0 && val<=15){
/*id_1st=id[x[0]][y[0]];*/              /* change 1st */
/*BitBlt_full();*/
/*printf_("id_1st=",id_1st,DX_,0);*/
/*printf(" new ID_1st=%d\n",id_1st);*/
}
else{                               /* id[][]=-2, -1 */
printf(" unpointed\n");
}

rectangle_(0,19*dlt,dlt-PPDY+2,(19+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
procedure=0;
}/**else if(procedure)**/
else if(procedure==2){
/*if(x[0]<0 || y[0]<0) printf(" ?\n");*/
if(x[0]>=0 && y[0]>=0 && (val=pixel[x[0]][y[0]])>=0 && val<=15){
/*id[x[0]][y[0]]=-1;*/                  /* restore one */
old=EDGE;EDGE=1;
search_i(x[0],y[0],16);
EDGE=old;
/*BitBlt_full();*/
}
else{
printf(" unpointed\n");
}
```

```
rectangle_(0,20*dlt,dlt-PPDY+2,(20+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
procedure=0;
}/**else if(procedure)**/
}/**if(Fill)****************************************************************/
else{
if(procedure==0){
BDflag=getflag(dx,dy);

if(BDflag==0 && xg==17 && clrpp==1){
rectangle_(0,xg*dlt,dlt-PPDY+2,(xg+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
}

if(BDflag==0){
if(xg==16){
restore_edge();
rectangle_(0,16*dlt,dlt-PPDY+2,(16+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
printf(" Restoration of edge\n");
}
else if(xg==17){
if(!clrpp) clrpp=1;else clrpp=0;
printf_("clrpp=",clrpp,DX_,2-1);
printf(" color PlusPlus\n");
}
else{
pcolor_old=pcolor;
pcolor=xg;
printf(" color=%d old=%d\n",xg,pcolor_old);
}
}/**if(BDflag)**/
else if(BDflag==1){
}/**else if(BDflag)**/
else if(BDflag==2){
     if(xg==0){
Fill=-1;
C_and_S(-1,-1,-1);
printf(" out of Fill(Fill:%d)\n",Fill);
}
else if(xg==1) procedure=1;          /* C */
else if(xg==2){                      /* AC */
Fill=0;
C_and_S(-1,-1,-1);
bdrnum=0;
rectangle_(0,(25+2)*dlt,dlt-PPDY+2,(25+3)*dlt,dlt-PPDY+3,bfset[WB].back,0);/* erase */
}
else if(xg==3){                      /* R */
/*refill=1;*/
fload("tmp.bin");
rectangle_(0,28*dlt,dlt-PPDY+2,(28+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
/*refill=2;*/
}
else if(xg==4){                      /* 48 */
#if HEX_or_SQR==1
if(!_4_or_8) _4_or_8=1;else _4_or_8=0;
printf_("_4_or_8=",_4_or_8,DX_,1-1);
printf(" 4_or_8\n");
rectangle_(0,29*dlt,dlt-PPDY+2,(29+1)*dlt,dlt-PPDY+3,bfset[WB].back,0);  /* erase */
#endif
}
}/**else if(BDflag)**/
else{
if(/*GKS_(ControlMask)*/x[0]>=0 && y[0]>=0){
if(Fill==0){
search_bdr(x[0],y[0]);
}
else{
if((i=bdrcheck(x[0],y[0]))<0) search_bdr(x[0],y[0]);
}
}/**if(GKS_(ControlMask))**/
}/**else(BDflag)**/
}/**if(procedure)**/
else if(procedure==1){
restore_magenta(x[0],y[0]);

rectangle_(0,(25+1)*dlt,dlt-PPDY+2,(25+2)*dlt,dlt-PPDY+3,bfset[WB].back,0);/* erase */
procedure=0;
}/**else if(procedure)**/
}/**else(Fill)****************************************************************/
}/**if(event.xbutton.button)**/
else{                                /* R_BUTTON */
if(Fill<0) return 1;
if(x[0]<0 || y[0]<0) return 1;

     if(Fill>0 && GKS_(ControlMask)<0 && GKS_(ShiftMask)>=0) val=0;   /* C+0 */
else if(Fill>0 && GKS_(ControlMask)>=0 && GKS_(ShiftMask)<0) val=1;   /* 0+S */
else if(Fill>0 && GKS_(ControlMask)>=0 && GKS_(ShiftMask)>=0) val=2;  /* C+S */
else if(Fill>0 && GKS_(ControlMask)<0 && GKS_(ShiftMask)<0) val=3;    /* C+S */
else if(Fill==0) val=4;                                               /* 0+S */
Fill=0;
xg=x[0];yg=y[0];
     if(val==0) {val=0;C_and_S(val,-1,pcolor);}  /* S, C */
else if(val==1) {val=1;C_and_S(val,pcolor,-1);}
else if(val==2) {val=2;C_and_S(val,pcolor,pcolor);}
else if(val==3) {val=2;C_and_S(val,pcolor,pcolor_old);}
else if(val==4) {val=3;C_and_S(val,pcolor,-1);}

bdrnum=0;
bdrsnum=0;SPmode=0;
/*xi=0;yi=0;*/                          /* for COPY==1 */
printf(" done\n");
}/**else(event.xbutton.button)**/

return 1;
}/**else if(event.type)**/
else if(event.type==MotionNotify){
if(YOUR_ART==0) return 1;
if(refill==1 || dialogflag>0/* || Fill==-1*/) return 1;
XQueryPointer(d,w,&root,&child,&rx,&ry,&wx,&wy,&mask);
if(!(mask & Button1Mask)) return 1;

dx=event.xbutton.x;dy=event.xbutton.y;
x[0]=dsp[dx][dy].cx;y[0]=dsp[dx][dy].cy;

if(x[0]<0 || y[0]<0) return 1;

if(Fill<0){
search_i(x[0],y[0],pcolor);
}/**if(Fill)**/
else{
if(Fill==0){
search_bdr(x[0],y[0]);
}
else{
if((i=bdrcheck(x[0],y[0]))<0) search_bdr(x[0],y[0]);
}
}/**else(Fill)**/

return 1;
}/**else if(event.type)**/

return 0;
}/** wndproc_filer **/
#endif


void write_bdr(char color)
{
int i;
```

```
for(i=0;i<bdrnum;i++) pixel_[bdr[i].cx][bdr[i].cy]=color;
}/** write_bdr **/


void write_bdrs(char color)
{
int i;

for(i=0;i<bdrsnum;i++)
pixel_[bdrs[i].cx][bdrs[i].cy]=color;
}/** write_bdrs **/


void C_and_S_old(char flag,int color,int color_old)
{
int i,pcolor,old,old_,val=2;
static int nest=0;

nest++;

if(flag==-1){                     /* restore all(magenta and white) */
old_=Fill;Fill=0;

for(i=0;i<bdrnum;i++){
val=pixel[bdr[i].cx][bdr[i].cy];
if(val==16){                      /* 16 first */
old=EDGE;EDGE=1;
search_i(bdr[i].cx,bdr[i].cy,16);
EDGE=old;
}
}/**for(i)**/

zi=1;
for(i=0;i<bdrnum;i++){
val=pixel[bdr[i].cx][bdr[i].cy];
if(val!=16){
search_i(bdr[i].cx,bdr[i].cy,val);
}
}/**for(i)**/
zi=0;

Fill=old_;
}
else if(flag==0){                 /* C+0 */
if(nest==1) fsave("tmp.bin");
old=c_trans;c_trans=2;searchflag=1;
```

```
for(i=0;i<bdrnum;i++)
search_i(bdr[i].cx,bdr[i].cy,color);

c_trans=old;searchflag=0;
}
else if(flag==1){                    /* 0+S */
fsave("tmp.bin");
old=c_trans;c_trans=1;

field(0);
write_bdr(0);
if(pixel_[xg][yg]==16){
val=cag_r(xg,yg,color);              /* 0:good, 1:bad */
if(val==0){
field(0);
c_trans=2;
write_bdr(0);
cag_r(xg,yg,color);
}

refill=2;
}/**if(pixel_[][])**/

c_trans=old;

C_and_S(-1,-1,-1);                   /* (-1,;restore all(magenta and white) */
}
else if(flag==2){                    /* C+S */
fsave("tmp.bin");
old=c_trans;c_trans=1;

field(0);
write_bdr(/*0*/-1);
if(pixel_[xg][yg]==16){
val=cag_r(xg,yg,color);              /* 0:good, 1:bad */
if(val==0){
field(0);
c_trans=2;
write_bdr(0);
cag_r(xg,yg,color);
}

refill=2;
}/**if(pixel_[][])**/
```

```
c_trans=old;

if(val==0)
C_and_S(0,color_old,-1);
else
C_and_S(-1,-1,-1);                    /* (-1,:restore all(magenta and white) */
}
else if(flag==3){                     /* S(no border(magenta and white)) */
fsave("tmp.bin");
old=c_trans;c_trans=1;

if((zg=pixel[xg][yg])!=color){
c_trans=2;
zeroFill=1;
cag_r(xg,yg,color);
zeroFill=0;

refill=2;
}/**if(pixel[][])**/

c_trans=old;

C_and_S(-1,-1,-1);                    /* (-1,;restore all(magenta and white) */
}

nest=0;
/*printf_("id_1st=",id_1st,DX_,0);*/
}/** C_and_S_old **/


void C_and_S(int flag,int color,int color_old)
{
int i,pcolor,old,old_,val,xCS,yCS;
static int nest=0;

xCS=xg;yCS=yg;
nest++;

if(flag==-1){                         /* restore all(magenta and white) */
old_=Fill;Fill=0;
predraw=0;

for(i=0;i<bdrnum;i++){
val=pixel[bdr[i].cx][bdr[i].cy];
if(val==16){                          /* 16 first */
old=EDGE;EDGE=1;
search_i(bdr[i].cx,bdr[i].cy,16);
EDGE=old;
}
}/**for(i)**/

zi=1;
for(i=0;i<bdrnum;i++){
val=pixel[bdr[i].cx][bdr[i].cy];
if(val!=16){
search_i(bdr[i].cx,bdr[i].cy,val);
}
}/**for(i)**/
zi=0;

predraw=1;
BitBlt_full();
Fill=old_;
}
else if(/*flag==0*/0){     /* 0:C_ */
if(nest==1) fsave("tmp.bin");
old=c_trans;

before:
printf(" C_:bdrnum:%d\n",bdrnum);

c_trans=1;SPmode=1;
field(0);                          /* 16 */
cag_r(xCS,yCS,color);              /* gets bdrs with CPMAX_;ca=16 */
printf(" C_:bdrsnum:%d\n",bdrsnum);



field(0);                          /* 16 */
c_trans=2;SPmode=0;
write_bdrs(/*0*/-1);  /* -1 */
searchflag=1;
tmp4=5;                            /* ca=-1 */
search_i_cag_r(xCS,yCS,-2,color_old);
tmp4=0;
searchflag=0;




refill=2;

c_trans=old;
```

```
C_and_S(-1,-1,-1);                  /* (-1,;restore all(magenta and white) */
}
else if(flag==0){     /* 0:C+0 */
fsave("tmp.bin");
old=c_trans;

c_trans=1;SPmode=0;
field(0);
write_bdr(/*0*/-1);
if(pixel_[xCS][yCS]==16){
printf(" C+0:bdrnum:%d\n",bdrnum);
#if HEX_or_SQR==0
val=cag_r_6(xCS,yCS,-1);            /* 0:good, 1:bad */
#else
if(_4_or_8==0)
val=cag_r_4(xCS,yCS,-1);                /* ca=zg */
else
val=cag_r_8(xCS,yCS,-1);                /* ca=zg */
#endif




if(val==0){
c_trans=1;SPmode=1;
field(0);
cag_r(xCS,yCS,color);               /* gets bdrs with CPMAX_ */
printf(" C+0:bdrsnum:%d\n",bdrsnum);

/*(1)*/
c_trans=1;SPmode=0;
field(0);
write_bdrs(/*0*/-1);
searchflag=1;
tmp4=2;                             /* ca=16 */
/*base=0;*/
search_i_cag_r(xCS,yCS,-2,color_old);
tmp4=0;
searchflag=0;

/*(3)*/
/*printf(" base:%d\n",base);
for(i=0;i<base;i++){
if(bdrs[i].clr==16) {old_=EDGE;EDGE=1;}
putpixel_noarray(bdrs[i].cx,bdrs[i].cy,bdrs[i].clr);
if(bdrs[i].clr==16) EDGE=old_;
}*/
```

```
}/**if(val)**/
else{
printf(" C+0 failed.\n");
}/**else(val)**/




refill=2;
}/**if(pixel_[][])**/
else if(pixel_[xCS][yCS]==-1) goto before;

c_trans=old;
C_and_S(-1,-1,-1);                  /* (-1,;restore all(magenta and white) */
}
else if(flag==1){     /* 1:0+S */
fsave("tmp.bin");
old=c_trans;

c_trans=1;SPmode=0;
field(0);
write_bdr(/*0*/-1);
printf(" 0+S:bdrnum:%d\n",bdrnum);
if(pixel_[xCS][yCS]==16){
#if HEX_or_SQR==0
val=cag_r_6(xCS,yCS,-1);            /* 0:good, 1:bad */
#else
if(_4_or_8==0)
val=cag_r_4(xCS,yCS,-1);                /* ca=zg */
else
val=cag_r_8(xCS,yCS,-1);                /* ca=zg */
#endif




if(val==0){
c_trans=1;SPmode=1;
field(0);
cag_r(xCS,yCS,color);               /* gets bdrs with CPMAX_ */
printf(" 0+S:bdrsnum:%d\n",bdrsnum);

/*(2)*/
field(0);
c_trans=2;SPmode=0;
write_bdrs(/*0*/-1);
searchflag=1;
tmp4=4;                             /* ca=16 */
```

```
search_i_cag_r(xCS,yCS,color,-2);
tmp4=0;
searchflag=0;
}/**if(val)**/
else{
printf(" 0+S failed.\n");
}/**else(val)**/



refill=2;
}/**if(pixel_[][])**/

c_trans=old;
C_and_S(-1,-1,-1);                  /* (-1,;restore all(magenta and white) */
}
else if(flag==2){     /* 2:C+S */
fsave("tmp.bin");
old=c_trans;

c_trans=1;SPmode=0;
field(0);
write_bdr(/*0*/-1);
printf(" C+S:bdrnum:%d\n",bdrnum);
if(pixel_[xCS][yCS]==16){
#if HEX_or_SQR==0
val=cag_r_6(xCS,yCS,-1);            /* 0:good, 1:bad */
#else
if(_4_or_8==0)
val=cag_r_4(xCS,yCS,-1);                /* ca=zg */
else
val=cag_r_8(xCS,yCS,-1);                /* ca=zg */
#endif



if(val==0){
c_trans=1;SPmode=1;
field(0);
cag_r(xCS,yCS,color);               /* gets bdrs with CPMAX_ */
printf(" C+S:bdrsnum:%d\n",bdrsnum);
/*for(i=0;i<bdrsnum;i++) putpixel_noarray(bdrs[i].cx,bdrs[i].cy,0);
use_subroop();*/

/*(1)*/
field(0);
```

```
c_trans=2;SPmode=0;
write_bdrs(/*0*/-1);
searchflag=1;
tmp4=2;                             /* ca=16 */
/*base=0;*/
color_old_g=color_old;
search_i_cag_r(xCS,yCS,color,-2);
val=rcount[0];
tmp4=0;
searchflag=0;

/*(3)*/
/*printf(" base:%d\n",base);
for(i=0;i<base;i++){
putpixel_noarray(bdrs[i].cx,bdrs[i].cy,color);

if(clrpp==1 && EDGE==0 && (i+1)%val==0){
if(color>-1) color++;if(color==16) color++;if(color>49) color=0;
}
}*/
}/**if(val)**/
else{
printf(" C+S failed.\n");
}/**else(val)**/




refill=2;
}/**if(pixel_[][])**/

c_trans=old;
C_and_S(-1,-1,-1);                  /* (-1,;restore all(magenta and white) */
}
else if(flag==3){     /* 3:S_(no border(magenta and white)) */
fsave("tmp.bin");
old=c_trans;c_trans=1;

/*999*/
if((zg=pixel[xCS][yCS])!=color/* && zg!=16*/){
tmp4=1;searchflag=1;                /* ca=16 */
c_trans=2;
zeroFill=1;
#if HEX_or_SQR==0
val=cag_r_6(xCS,yCS,-1);                /* ca=zg */
#else
if(_4_or_8==0)
```

```
val=cag_r_4(xCS,yCS,-1);                /* ca=zg */
else
val=cag_r_8(xCS,yCS,-1);                /* ca=zg */
#endif
zeroFill=0;
tmp4=0;searchflag=0;



printf(" S_:bdrnum:%d\n",bdrnum);

if(val==0){
c_trans=1;SPmode=1;
field(0);                           /* 16 */
cag_r(xCS,yCS,color);               /* gets bdrs with CPMAX_;ca=16 */
printf(" S_:bdrsnum:%d\n",bdrsnum);

field(0);                           /* 16 */
c_trans=2;SPmode=0;
write_bdrs(/*0*/-1);  /* -1 */
searchflag=1;
tmp4=3;                             /* ca=-1 */
search_i_cag_r(xCS,yCS,color,-2);
tmp4=0;
searchflag=0;
}/**if(val)**/
else{
printf(" S_ failed.\n");
fload("tmp.bin");
}/**else(val)**/



refill=2;
}/**if(pixel[][])**/

c_trans=old;
}

nest=0;
/*printf_("id_1st=",id_1st,DX_,0);*/
}/** C_and_S **/


int bdrcheck(int x,int y)
{
```

```
int i;

for(i=0;i<bdrnum;i++){
if(x==bdr[i].cx && y==bdr[i].cy) return i;
}

return -1;
}/** bdrcheck **/


void delay_(long millisecond)
{
long oldtime,nowtime,dtime,old;
dbl 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;

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


void beep(long millisecond)
{
#if WX==0
Beep(888,millisecond);
#endif
}/** beep **/


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


void restore_work(char flag)
{
int i;

if(flag==0){
if(idx){
for(i=idx-1;i>-1;i--){
work[stack[i].x][stack[i].y].p=0;
}

idx=0;
}
}
else{
/*if(idx_Rect){
for(i=idx_Rect-1;i>-1;i--){
work_Rect[stack_Rect[i].x][stack_Rect[i].y]=0;
}

idx_Rect=0;
}*/
}
}/** restore_work **/


#if WX==0
int ppixel(int nx,int ny,int pcolor)
{
if((nx<0)||(nx>XRESO-1)||(ny<0-PPDY)||(ny>YRESO-1)) return 1;
/*if((nx<0)||(nx>XRESO-1)||(ny<0)||(ny>YrESO-1)) return 1;*/

ny+=PPDY;
```

```
if(predraw>0) SetPixelV(hdcdisplay,nx,ny,PALETTE(pcolor));
if(1){
if(GRPH) SetPixelV(hdctmp1,nx,ny,PALETTE(pcolor));
if(fieldflag==0 && c_trans==0){
dsp[nx][ny].cx=xi;dsp[nx][ny].cy=yi;  /* nx, ny:display coordinates */
}
}
else{
/*SetPixelV(hdctmp1,nx,ny,PALETTE(pcolor));
if(xg==0){
SetPixelV(hdctmp1,nx+1,ny,PALETTE(pcolor));
SetPixelV(hdctmp1,nx-1,ny,PALETTE(pcolor));
}
else{
SetPixelV(hdctmp1,nx,ny+1,PALETTE(pcolor));
SetPixelV(hdctmp1,nx,ny-1,PALETTE(pcolor));
}*/
}

return 0;
}/** ppixel **/
#else
int ppixel(int nx,int ny,int pcolor)
{
if((nx<0)||(nx>XRESO-1)||(ny<0-PPDY)||(ny>YRESO-1)) return 1;
/*if((nx<0)||(nx>XRESO-1)||(ny<0)||(ny>YrESO-1)) return 1;*/

ny+=PPDY;
XSetForeground(d,gcdisplay,irgb[pcolor].pixel);

if(predraw>0) XDrawPoint(d,w,gcdisplay,nx,ny);
if(1){
if(GRPH) XDrawPoint(d,pmap1,gcdisplay,nx,ny);
if(fieldflag==0 && c_trans==0){
dsp[nx][ny].cx=xi;dsp[nx][ny].cy=yi;  /* nx, ny:display coordinates */
}
}
else{
/*SetPixelV(hdctmp1,nx,ny,PALETTE(pcolor));
if(xg==0){
SetPixelV(hdctmp1,nx+1,ny,PALETTE(pcolor));
SetPixelV(hdctmp1,nx-1,ny,PALETTE(pcolor));
}
else{
SetPixelV(hdctmp1,nx,ny+1,PALETTE(pcolor));
SetPixelV(hdctmp1,nx,ny-1,PALETTE(pcolor));
}*/
}

return 0;
}/** ppixel **/
#endif


void hline(int left,int right,int y,int color)
{
int i,xs,ys;
dbl DX,DY,Dz,X,Y,Z,len,val;

DX=work[right][y].x-work[left][y].x;
DY=work[right][y].y-work[left][y].y;
Dz=work[right][y].z-work[left][y].z;

for(i=left+1;i<=right-1;i++){
xs=i+dO[0]-Zx/2;
ys=y+dO[1]-Zy/2;

if(/*Zflag==0*/1) ppixel(xs,ys,color);
else{
}
}/**for(i)**/
}/** hline **/


void line(dbl x1_,dbl y1_,dbl x2_,dbl y2_,int color)
{
int x1,y1,x2,y2,dx,dy,x,y;
int c,d,e,sx,sy;
int putflag=1,putcount=0;

x1=ff_fc(x1_);y1=ff_fc(y1_);
x2=ff_fc(x2_);y2=ff_fc(y2_);

dx=abs(x2-x1);
dy=abs(y2-y1);

if(x1<=x2) sx=1;
else       sx=-1;
if(y1<=y2) sy=1;
else       sy=-1;
```

```
if(dx>=dy){
c=2*dy;d=2*(dy-dx);e=c-dx;

x=x1;
y=y1;

while(1){
if(putperiod==0) ;
else{
if(putcount==putperiod){
putcount=0;
if(putflag==1) putflag=0;else putflag=1;
}
putcount++;
}

if(putperiod==0 || putflag==1) ppixel(x,y,color);

if(e<0) e+=c;
else    {e+=d;y+=sy;}

x+=sx;
if(sx>=0) {if(x>x2) break;}
else      {if(x<x2) break;}
}
}/**if(dx,dy)**/
else{
c=2*dx;d=2*(dx-dy);e=c-dy;

x=x1;
y=y1;

while(1){
if(putperiod==0) ;
else{
if(putcount==putperiod){
putcount=0;
if(putflag==1) putflag=0;else putflag=1;
}
putcount++;
}

if(putperiod==0 || putflag==1) ppixel(x,y,color);

if(e<0) e+=c;
else    {e+=d;x+=sx;}
```

```
y+=sy;
if(sy>=0) {if(y>y2) break;}
else      {if(y<y2) break;}
}
}/**else(dx,dy)**/
}/** line **/


void line_eye_(dbl xd1,dbl yd1,dbl zd1,dbl xd2,dbl yd2,dbl zd2,int color)
{
int x1,y1,x2,y2,dx,dy,x,y;
int c,d,e,sx,sy;
int putflag=1,putcount=0;
int xb,yb;
dbl DX,DY,Dz,X,Y,Z,len,val,tmp0,tmp1,tmp2;

/*projection(xd1,yd1,zd1,&x1,&y1);
projection(xd2,yd2,zd2,&x2,&y2);*/
x1=xd1;y1=yd1;
x2=xd2;y2=yd2;

dx=x2-x1;
if(dx<0){
dx=x1;dy=y1;
x1=x2;y1=y2;
x2=dx;y2=dy;

tmp0=xd1;tmp1=yd1;tmp2=zd1;
xd1=xd2;yd1=yd2;zd1=zd2;
xd2=tmp0;yd2=tmp1;zd2=tmp2;
}

DX=xd2-xd1;
DY=yd2-yd1;
Dz=zd2-zd1;



dx=abs(x2-x1);
dy=abs(y2-y1);

if(dx==0 && dy==0){
/* x1,y1 */
if(Zflag==0) ppixel(x1,y1,color);
else{
```

```
xb=x1-dO[0]+Zx/2;
yb=y1-dO[1]+Zy/2;
if(xb<0 || xb>Zx-1 || yb<0 || yb>Zy-1) ;
else{
if(/*len<Zbuf[Zpage][xb][yb] || Zflag==2*/1){
/*if(color==0){*/
ppixel(x1,y1,color);
/*}
else{
if(work_Rect[xb][yb]==0) {ppixel(x1,y1,color);work_Rect[xb][yb]=2;}
}*/
}
/*else if(color==0 && work_Rect[xb][yb]==2) ppixel(x1,y1,color);*/

stack[idx].x=xb;stack[idx].y=yb;idx++;
/*if(color==0) work_Rect[xb][yb]=1;
stack_Rect[idx_Rect].x=xb;stack_Rect[idx_Rect].y=yb;idx_Rect++;*/
work[xb][yb].p=1;
work[xb][yb].x=X;
work[xb][yb].y=Y;
work[xb][yb].z=Z;
}
}

return;
}

if(x1<=x2) sx=1;
else       sx=-1;
if(y1<=y2) sy=1;
else       sy=-1;

x=x1;
y=y1;

if(dx>=dy){
c=2*dy;d=2*(dy-dx);e=c-dx;

while(1){
if(putperiod==0) ;
else{
if(putcount==putperiod){
putcount=0;
if(putflag==1) putflag=0;else putflag=1;
}
putcount++;
}

if(putperiod==0 || putflag==1) ;else goto next_dx;



if(Zflag==0) ppixel(x,y,color);
else{
xb=x-d0[0]+Zx/2;
yb=y-d0[1]+Zy/2;
if(xb<0 || xb>Zx-1 || yb<0 || yb>Zy-1) ;
else{
if(/*len<Zbuf[Zpage][xb][yb] || Zflag==2*/1){
/*if(color==0){*/
ppixel(x,y,color);
/*}
else{
if(work_Rect[xb][yb]==0) {ppixel(x,y,color);work_Rect[xb][yb]=2;}
}*/
}
/*else if(color==0 && work_Rect[xb][yb]==2) ppixel(x,y,color);*/

stack[idx].x=xb;stack[idx].y=yb;idx++;
/*if(color==0) work_Rect[xb][yb]=1;
stack_Rect[idx_Rect].x=xb;stack_Rect[idx_Rect].y=yb;idx_Rect++;*/
work[xb][yb].p=1;
work[xb][yb].x=X;
work[xb][yb].y=Y;
work[xb][yb].z=Z;
}
}



next_dx:

if(e<0) e+=c;
else    {e+=d;y+=sy;}

x+=sx;
if(sx>=0) {if(x>x2) break;}
else      {if(x<x2) break;}
}
}/**if(dx,dy)**/
else{
c=2*dx;d=2*(dx-dy);e=c-dy;
```

```
while(1){
if(putperiod==0) ;
else{
if(putcount==putperiod){
putcount=0;
if(putflag==1) putflag=0;else putflag=1;
}
putcount++;
}

if(putperiod==0 || putflag==1) ;else goto next_dy;



if(Zflag==0) ppixel(x,y,color);
else{
xb=x-dO[0]+Zx/2;
yb=y-dO[1]+Zy/2;
if(xb<0 || xb>Zx-1 || yb<0 || yb>Zy-1) ;
else{
if(/*len<Zbuf[Zpage][xb][yb] || Zflag==2*/1){
/*if(color==0){*/
ppixel(x,y,color);
/*}
else{
if(work_Rect[xb][yb]==0) {ppixel(x,y,color);work_Rect[xb][yb]=2;}
}*/
}
/*else if(color==0 && work_Rect[xb][yb]==2) ppixel(x,y,color);*/

stack[idx].x=xb;stack[idx].y=yb;idx++;
/*if(color==0) work_Rect[xb][yb]=1;
stack_Rect[idx_Rect].x=xb;stack_Rect[idx_Rect].y=yb;idx_Rect++;*/
work[xb][yb].p=1;
work[xb][yb].x=X;
work[xb][yb].y=Y;
work[xb][yb].z=Z;
}
}



next_dy:

if(e<0) e+=c;
else    {e+=d;x+=sx;}

y+=sy;
if(sy>=0) {if(y>y2) break;}
else      {if(y<y2) break;}
}
}/**else(dx,dy)**/
}/** line_eye_ **/


void Trian(char flag,dbl x1,dbl y1,dbl z1,dbl x2,dbl y2,dbl z2,
           dbl x3,dbl y3,dbl z3,int color1,int color2,int color3,int color)
{
int i;
int ymin,ymax,y,xmin,xmax,xmin_,xmax_;

restore_work(0);
/*if(Rectflag==0) restore_work(1);*/

if(flag==0){
line_eye_(x1,y1,-1,x2,y2,-1,color3);
line_eye_(x2,y2,-1,x3,y3,-1,color1);
line_eye_(x3,y3,-1,x1,y1,-1,color2);
}
else{
line_eye_(x1,y1,-1,x2,y2,-1,color3);
line_eye_(x2,y2,-1,x3,y3,-1,color1);
line_eye_(x3,y3,-1,x1,y1,-1,color2);
}

if(idx==0) return;

i=0;ymin=Zy;
while(1){
if(stack[i].y<ymin) ymin=stack[i].y;

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

i=0;ymax=-1;
while(1){
if(stack[i].y>ymax) ymax=stack[i].y;

i++;if(i==idx) break;
}
```

```
for(y=ymin;y<=ymax;y++){
i=0;xmin=Zx;
while(1){
if(stack[i].y==y && stack[i].x<xmin) xmin=stack[i].x;

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

i=0;xmax=-1;
while(1){
if(stack[i].y==y && stack[i].x>xmax) xmax=stack[i].x;

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

i=xmin;
while(1){
if(work[i+1][y].p==0) {xmin_=i;break;}

i++;if(i==Zx-1) break;
}

i=xmax;
while(1){
if(work[i-1][y].p==0) {xmax_=i;break;}

i--;if(i==0) break;
}

if(xmax_>=xmin_+2){
hline(xmin_,xmax_,y,color);
}
}/**for(y)**/
}/** Trian **/


void Polyline_(POINT *vertex,int num,int pencolor)
{
int i,old;

old=xg;

for(i=0;i<num;i++){
if(vertex[i].x==vertex[i+1].x) xg=0;else xg=1;
line(vertex[i].x,vertex[i].y,vertex[i+1].x,vertex[i+1].y,pencolor);
}
```

```
xg=old;
}/** Polyline_ **/


void Polygon_(POINT *vertex,int num,int color)
{
int i,x,y;

/*return;*/

if(num==4){
Trian(0,vertex[0].x,vertex[0].y,-1,vertex[1].x,vertex[1].y,-1,
        vertex[2].x,vertex[2].y,-1,color,color,color,color);
Trian(0,vertex[0].x,vertex[0].y,-1,vertex[2].x,vertex[2].y,-1,
        vertex[3].x,vertex[3].y,-1,color,color,color,color);
}
else if(num==6){
x=(vertex[0].x+vertex[3].x)/2;
y=(vertex[0].y+vertex[3].y)/2;
for(i=0;i<num;i++)
Trian(0,vertex[i].x,vertex[i].y,-1,vertex[i+1].x,vertex[i+1].y,-1,
        x,y,-1,color,color,color,color);
}
}/** Polygon_ **/


void mem_bdr(int x,int y)
{
int i;

if((i=bdrcheck(x,y))<0){
Fill=1;
bdr[bdrnum].cx=x;bdr[bdrnum].cy=y;
printf(" mem_bdr(%d,%d)\n",bdr[bdrnum].cx,bdr[bdrnum].cy);
bdrnum++;
}
}/** mem_bdr **/


void search_bdr(int x,int y)
{
Fill=1;
bdr[bdrnum].cx=x;bdr[bdrnum].cy=y;
search_i(x,y,-1);
/*printf(" %d %d\n",bdr[bdrnum].cx,bdr[bdrnum].cy);*/
```

```
bdrnum++;

/*bdrflag=1;
putpixel_(x,y,-1);*/
if(0) mem_bdr(x,y);                          /* c_trans=3, Fill=1 */
/*bdrflag=0;*/

/*if(c_trans==3 && Fill>0) putpixels_bdr_out(0,x,y);*/
}/** search_bdr **/


void restore_magenta(int xr,int yr)
{
char hit=0;
int i,i_,j,k,val,old_,old;

/* restore one */
for(i=0;i<bdrnum;i++){
if(bdr[i].cx==xr && bdr[i].cy==yr){
old_=Fill;Fill=0;
val=pixel[bdr[i].cx][bdr[i].cy];    /* restore one(magenta and white) */
if(val!=16)
search_i(bdr[i].cx,bdr[i].cy,val);
else{
old=EDGE;EDGE=1;
search_i(bdr[i].cx,bdr[i].cy,val);
EDGE=old;
}
Fill=old_;

for(k=i+1;k<bdrnum;k++){
bdr[k-1].cx=bdr[k].cx;
bdr[k-1].cy=bdr[k].cy;
}
bdrnum--;hit=1;



/*if(0){
search_bdr_flag=1;

ig=0;
putpixel_(xr,yr,-1);

for(j=0;j<MAX_VTX;j++)
if(nx_g[ig][j]>=0){
```

```
for(i_=0;i_<bdrnum;i_++){
if(bdr[i_].cx==nx_g[ig][j] && bdr[i_].cy==ny_g[ig][j]){
printf(" MAX_VTX\n");

old_=Fill;Fill=0;
val=pixel[bdr[i_].cx][bdr[i_].cy];
if(val!=16)
search_i(bdr[i_].cx,bdr[i_].cy,val);
else{
old=EDGE;EDGE=1;
search_i(bdr[i_].cx,bdr[i_].cy,val);
EDGE=old;
}
Fill=old_;

for(k=i_+1;k<bdrnum;k++){
bdr[k-1].cx=bdr[k].cx;
bdr[k-1].cy=bdr[k].cy;
}
bdrnum--;

break;
}
}
}

search_bdr_flag=0;
}*/



/*BitBlt_full();*/
break;                              /* for(i) */
}/**if(bdr[i].cx,bdr[i].cy)**/
}/**for(i)**/

if(!hit) printf(" restore_magenta:unpointed\n");
}/** restore_magenta **/


int getflag(int x,int y)
{
int i,val,dlt=SQSZ;

val=3;
```

```
for(i=0;i<0+18;i++)                 /* left */
if(x>i*dlt && x<(i+1)*dlt && y>0/*-PPDY*/ && y<dlt/*-PPDY*/){
xg=i-0;
     if(xg==16) val=-1;             /* R */
else if(xg==17) val=-2;             /* PP */
else            val=0;              /* 16 colors */
break;
}

if(val==3 && Fill==-1)              /* centre */
for(i=19;i<19+5;i++)
if(x>i*dlt && x<(i+1)*dlt && y>0/*-PPDY*/ && y<dlt/*-PPDY*/) {val=1;xg=i-19;break;}

if(val==3)                          /* right */
for(i=25;i<25+/*4*/5;i++)
if(x>i*dlt && x<(i+1)*dlt && y>0/*-PPDY*/ && y<dlt/*-PPDY*/) {val=2;xg=i-25;break;}

if(val<0){                          /* R, PP */
rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
/*BitBlt_full();*/
}
else if(val==0){                    /* 16 colors */
rectangle_(0,0*dlt,dlt-PPDY+2,(0+16)*dlt,dlt-PPDY+3,bfset[WB].back,i);  /* erase */
/*if(i==17){
if(clrpp==0) rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
}
else*/
rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
/*BitBlt_full();*/
}
else if(val==1){                    /* centre */
rectangle_(0,19*dlt,dlt-PPDY+2,(19+5)*dlt,dlt-PPDY+3,bfset[WB].back,i);  /* erase */
rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
/*BitBlt_full();*/
}
else if(val==2){                    /* right */
if(Fill>-1 && xg==0)
rectangle_(0,25*dlt,dlt-PPDY+2,(25+1)*dlt,dlt-PPDY+3,bfset[WB].back,i);  /* erase */
else if(Fill==-1 && xg==0)
rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
else if(Fill>-1 && xg==1)
rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
else if(Fill>-1 && xg==2)
rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
else if(Fill>-1 && xg==3)
rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
```

```
else if(Fill>-1 && xg==4)
rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
/*BitBlt_full();*/
}

if(val<0) return 0;
else      return val;
}/** getflag **/


#if HEX_or_SQR==0
void search_i(int cx,int cy,int pcolor)
{
char flag=0,str[32];
int i,j,k,cnt,cnt_,id_tmp,j_,the_color,upper;

/*bdrcolor=0;*/

for(i=0;i<RCMAX;i++)
for(j=0;j<CPMAX_;j++)
for(k=0;k<10;k++){
if(mbr[i].x[j][k]<0) break;

if(mbr[i].x[j][k]==cx && mbr[i].y[j][k]==cy){
/*id_tmp=id_1st;*/
if(1){
/*printf_("id_1st=",id_1st,DX_,0);*/
/*printf_("ig    =",j,DX_,1-1);*/
}

cnt=0;

if(/*CPHALF==CPMAX*/0){
}/**if(CPHALF)**/
else if(/*CPHALF==CPMAX/2*/0){
}/**else if(CPHALF)**/
else{
upper=(j/CPGRP+1)*CPGRP-1;
yg=id_tmp;

begin_:
cnt_=0;
j_=j;

while(1){
/* c_trans=3 */
xg=id_tmp;
if(zi==0)
putpixel_(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor);
else{
the_color=pixel[mbr[i].x[j_][0]][mbr[i].y[j_][0]];
putpixel_(mbr[i].x[j_][0],mbr[i].y[j_][0],the_color);
}
/*printf(" %d %d\n",mbr[i].x[j_][0],mbr[i].y[j_][0]);*/

j_++;
if(j_>upper){
j_-=CPGRP;
}

id_tmp++;if(id_tmp==CPMAX_) id_tmp=0;

cnt++;
if(cnt==CPMAX_) break;
if(clrpp==1 && EDGE==0){
pcolor++;if(pcolor==16) pcolor++;if(pcolor>49) pcolor=0;
}

cnt_++;
if(cnt_==CPGRP){
j+=CPGRP;upper+=CPGRP;
if(j>CPMAX_-1) {j-=CPMAX_;upper-=CPMAX_;}
goto begin_;
}

}/**while(1)**/
}/**else(CPHALF)**/

flag=1;
/*BitBlt_full();*/
goto end;
}/**if(mbr[][], mbr[][])**/
}

end:
/*printf(" search:%d %d\n",cx,cy)*/;
}/** search_i **/
#else
void search_i(int cx,int cy,int pcolor)
{
char flag=0,str[32];
int i,j,k,cnt,cnt_,id_tmp,j_,the_color;
```

```
/*#if COPY
search_c(cx,cy,pcolor);
return;
#endif*/

for(i=0;i<RCMAX;i++)
for(j=0;j<CPMAX;j++)
for(k=0;k<10;k++){
if(mbr[i].x[j][k]<0) break;

if(mbr[i].x[j][k]==cx && mbr[i].y[j][k]==cy){
/*id_tmp=id_1st;*/
if(1){
/*printf_("id_1st=",id_1st,DX_,0);*/
/*printf_("ig    =",j,DX_,1-1);*/
}

cnt=0;

if(CPHALF==CPMAX){
j_=j;
yg=id_tmp;

while(1){
/* c_trans=3 */
xg=id_tmp;
if(zi==0)
putpixel_(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor);
else{
the_color=pixel[mbr[i].x[j_][0]][mbr[i].y[j_][0]];
putpixel_(mbr[i].x[j_][0],mbr[i].y[j_][0],the_color);
}
/*printf(" %d %d\n",mbr[i].x[j_][0],mbr[i].y[j_][0]);*/

j_++;if(j_==CPMAX_) j_=0;
id_tmp++;if(id_tmp==CPMAX_) id_tmp=0;

cnt++;if(cnt==CPMAX_) break;
if(clrpp==1 && EDGE==0){
pcolor++;if(pcolor==16) pcolor++;if(pcolor>49) pcolor=0;
}
}/**while(1)**/
}/**if(CPHALF)**/
else{
}/**else(CPHALF)**/
```

```
flag=1;
/*BitBlt_full();*/
goto end;
}/**if(mbr[][], mbr[][])**/
}

end:
/*printf(" search:%d %d\n",cx,cy)*/;
}/** search_i **/
#endif


#if HEX_or_SQR==1
void search_c(int cx,int cy,int pcolor)
{
int x,y,dx,cnt_x,cnt_y;

dx=2*Lsqr+1;
yg=0;

xg=0;
y=cy;

/* down */
cnt_y=0;

while(1){
cnt_x=0;
putpixel_(cx,y,pcolor);xg++;cnt_x++;cnt_y++;
if(clrpp==1 && EDGE==0) {pcolor++;if(pcolor>15) pcolor=0;}

/* right */
x=cx;
while(1){
if(x!=cx){
putpixel_(x,y,pcolor);xg++;cnt_x++;
if(clrpp==1 && EDGE==0) {pcolor++;if(pcolor>15) pcolor=0;}
}

if(xi>0 && cnt_x==cpwidth-xi+1) break;
else if(xi<0 && cnt_x==cpwidth-(-xi)+1) break;

x+=dx;
if(x>RESO-1){
if(xi==0) break;
else x-=RESO;
}
}

/* left */
x=cx;cnt_x=0;
while(1){
if(x!=cx){
putpixel_(x,y,pcolor);xg++;cnt_x++;
if(clrpp==1 && EDGE==0) {pcolor++;if(pcolor>15) pcolor=0;}
}

if(xi>0 && cnt_x==xi-1) break;
else if(xi<0 && cnt_x==(-xi)-1) break;

x-=dx;if(x<0) break;
}

if(yi>0 && cnt_y==cpwidth-yi+1) break;
else if(yi<0 && cnt_y==cpwidth-(-yi)+1) break;

y+=dx;
if(y>RESO-1){
if(yi==0) break;
else y-=RESO;
}
}



y=cy;
/*y-=dx;if(y<0) goto end;*/

/* up */
cnt_y=0;

while(1){
if(yi>0 && cnt_y==yi-1) break;
else if(yi<0 && cnt_y==(-yi)-1) break;

y-=dx;if(y<0) break;

cnt_x=0;
putpixel_(cx,y,pcolor);xg++;cnt_x++;cnt_y++;
if(clrpp==1 && EDGE==0) {pcolor++;if(pcolor>15) pcolor=0;}
```

```
/* right */
x=cx;
while(1){
if(x!=cx){
putpixel_(x,y,pcolor);xg++;cnt_x++;
if(clrpp==1 && EDGE==0) {pcolor++;if(pcolor>15) pcolor=0;}
}

if(xi>0 && cnt_x==cpwidth-xi+1) break;
else if(xi<0 && cnt_x==cpwidth-(-xi)+1) break;

x+=dx;
if(x>RESO-1){
if(xi==0) break;
else x-=RESO;
}
}

/* left */
x=cx;cnt_x=0;
while(1){
if(x!=cx){
putpixel_(x,y,pcolor);xg++;cnt_x++;
if(clrpp==1 && EDGE==0) {pcolor++;if(pcolor>15) pcolor=0;}
}

if(xi>0 && cnt_x==xi-1) break;
else if(xi<0 && cnt_x==(-xi)-1) break;

x-=dx;if(x<0) break;
}

/*if(yi>0 && cnt_y==yi-1) break;
else if(yi<0 && cnt_y==(-yi)-1) break;*/

/*y-=dx;if(y<0) break;*/
}

end:
;
}/** search_c **/
#endif


#if HEX_or_SQR==0
void search_i_cag_r(int cx,int cy,int pcolor,int pcolor_old)
```

```
{
char flag=0,str[32];
int i,j,k,cnt,cnt_,id_tmp,j_,the_color,upper;

for(i=0;i<RCMAX;i++)
for(j=0;j<CPMAX_;j++)
for(k=0;k<10;k++){
if(mbr[i].x[j][k]<0) break;

if(mbr[i].x[j][k]==cx && mbr[i].y[j][k]==cy){
/*id_tmp=id_1st;*/
if(1){
/*printf_("id_1st=",id_1st,DX_,0);*/
/*printf_("ig    =",j,DX_,1-1);*/
}

cnt=0;

if(/*CPHALF==CPMAX*/0){
}/**if(CPHALF)**/
else if(/*CPHALF==CPMAX/2*/0){
}/**else if(CPHALF)**/
else{
upper=(j/CPGRP+1)*CPGRP-1;
yg=id_tmp;

begin_:
cnt_=0;
j_=j;

while(1){
/* c_trans=2 */
xg=id_tmp;
     if(tmp4==2){
     if(c_trans==1) cag_r_6(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor_old);
else if(c_trans==2) cag_r_6(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor);
}
else if(tmp4==3 || tmp4==4){
if(/*tmp4==3 && pixel_[mbr[i].x[j_][0]][mbr[i].y[j_][0]]!=-1*/0) ;
else cag_r_6(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor);
}
else if(tmp4==5){
cag_r_6(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor_old);
}

j_++;
if(j_>upper){
j_-=CPGRP;
}

id_tmp++;if(id_tmp==CPMAX_) id_tmp=0;

cnt++;
if(cnt==CPMAX_) break;
if(clrpp==1 && EDGE==0){
if(pcolor>-1) pcolor++;if(pcolor==16) pcolor++;if(pcolor>49) pcolor=0;
if(pcolor_old>-1) pcolor_old++;
if(pcolor_old==16) pcolor_old++;if(pcolor_old>49) pcolor_old=0;
if(color_old_g>-1) color_old_g++;
if(color_old_g==16) color_old_g++;if(color_old_g>49) color_old_g=0;
}

cnt_++;
if(cnt_==CPGRP){
j+=CPGRP;upper+=CPGRP;
if(j>CPMAX_-1) {j-=CPMAX_;upper-=CPMAX_;}
goto begin_;
}

}/**while(1)**/
}/**else(CPHALF)**/

flag=1;
/*BitBlt_full();*/
goto end;
}/**if(mbr[][], mbr[][])**/
}

end:
/*printf(" search:%d %d\n",cx,cy)*/;
}/** search_i_cag_r **/
#else
void search_i_cag_r(int cx,int cy,int pcolor,int pcolor_old)
{
char flag=0,str[32];
int i,j,k,cnt,cnt_,id_tmp,j_,the_color;

/*#if COPY
search_c(cx,cy,pcolor);
return;
#endif*/
```

```
for(i=0;i<RCMAX;i++)
for(j=0;j<CPMAX;j++)
for(k=0;k<10;k++){
if(mbr[i].x[j][k]<0) break;

if(mbr[i].x[j][k]==cx && mbr[i].y[j][k]==cy){
/*id_tmp=id_1st;*/
if(1){
/*printf_("id_1st=",id_1st,DX_,0);*/
/*printf_("ig    =",j,DX_,1-1);*/
}

cnt=0;

if(/*CPHALF==CPMAX*/1){
j_=j;
yg=id_tmp;

while(1){
/* c_trans=3 */
xg=id_tmp;
     if(tmp4==2){
     if(c_trans==1){
if(_4_or_8==0)
cag_r_4(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor_old);
else
cag_r_8(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor_old);
}
else if(c_trans==2){
if(_4_or_8==0)
cag_r_4(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor);
else
cag_r_8(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor);
}
}
else if(tmp4==3 || tmp4==4){
if(/*tmp4==3 && pixel_[mbr[i].x[j_][0]][mbr[i].y[j_][0]]!=-1*/0) ;
else{
if(_4_or_8==0)
cag_r_4(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor);
else
cag_r_8(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor);
}/**else(tmp4, pixel_[mbr[i].x[j_][0]][mbr[i].y[j_][0]])**/
}
else if(tmp4==5){
cag_r_8(mbr[i].x[j_][0],mbr[i].y[j_][0],pcolor_old);
}

j_++;if(j_==CPMAX_) j_=0;
id_tmp++;if(id_tmp==CPMAX_) id_tmp=0;

cnt++;if(cnt==CPMAX_) break;
if(clrpp==1 && EDGE==0){
if(pcolor>-1) pcolor++;if(pcolor==16) pcolor++;if(pcolor>49) pcolor=0;
if(pcolor_old>-1) pcolor_old++;
if(pcolor_old==16) pcolor_old++;if(pcolor_old>49) pcolor_old=0;
if(color_old_g>-1) color_old_g++;
if(color_old_g==16) color_old_g++;if(color_old_g>49) color_old_g=0;
}
}/**while(1)**/
}/**if(CPHALF)**/
else{
}/**else(CPHALF)**/

flag=1;
/*BitBlt_full();*/
goto end;
}/**if(mbr[][], mbr[][])**/
}

end:
/*printf(" search:%d %d\n",cx,cy)*/;
}/** search_i_cag_r **/
#endif

/************************** dialog functions -> ***************************/

#if WX==0
void setcsrcolor(int color)
{
hbrush=CreateSolidBrush(PALETTE(color));
SelectObject(hdctmp3,hbrush);
PatBlt(hdctmp3,0,0,XRESO,/*YRESO*/UDY,PATCOPY);
DeleteObject(hbrush);
}/** setcsrcolor **/
#else
void setcsrcolor(int color)
{
XSetForeground(d,gcdisplay,irgb[color].pixel);
XFillRectangle(d,pmap3,gcdisplay,0,0,XRESO,UDY);  /* for cursor */
}/** setcsrcolor **/
#endif
```

```
void overwrite(void)
{
if(insorover==0) return;

overwriteflag=1;

if(dialogflag>0){
lumpflag_dialog=1;
deletion_dialog();
lumpflag_dialog=0;
}
/*else
deletion_onlymem();*/

overwriteflag=0;
}/** overwrite **/


#if WX==0
void imm_check(void)
{
himc_=ImmGetContext(hwnd);
if(immflag==1 && ImmGetOpenStatus(himc_)==TRUE) immflag=/*0*/2;
ImmReleaseContext(hwnd,himc_);
}/** imm_check **/


void imm_pause(void)
{
himc_=ImmGetContext(hwnd);
if(ImmGetOpenStatus(himc_)==TRUE){
immflag=1;ImmSetOpenStatus(himc_,FALSE);imm_restart_flag=1;
}
else imm_restart_flag=0;
ImmReleaseContext(hwnd,himc_);
}/** imm_pause **/


void imm_restart(void)
{
himc_=ImmGetContext(hwnd);
if(immflag==1 && ImmGetOpenStatus(himc_)==FALSE){
immflag=/*0*/2;ImmSetOpenStatus(himc_,TRUE);
}
```

```
else immflag=0;
/*imm_restart_flag=0;*/                 /* no problem ? */
ImmReleaseContext(hwnd,himc_);
}/** imm_restart **/


void imm_close(void)
{
himc_=ImmGetContext(hwnd);
if(ImmGetOpenStatus(himc_)==TRUE) ImmSetOpenStatus(himc_,FALSE);
immflag=0;
ImmReleaseContext(hwnd,himc_);
}/** imm_close **/


void breaks(char flag)
{
/*extraline(0);*/cqflag=0;
if(flag==1){
charflag=0;charcode=2;
}
}/** breaks **/
#else
void imm_pause(void)
{
XUnsetICFocus(ic);
imm_restart_flag=1;
}/** imm_pause **/


void imm_restart(void)
{
XSetICValues(ic,XNFocusWindow,w,NULL);
XSetICFocus(ic);
InputPosition(ic,icsr,jcsr);
/*imm_restart_flag=0;*/                 /* no problem ? */
}/** imm_restart **/
#endif


#if WX==0
void WM_func_CHAR(WPARAM wparam)
{
unsigned char charcode_tmp;

BitBltflag=0;BitBltflag_=0;
```

```
charcode_tmp=(unsigned char)wparam;    /* bridge */
/*if(charcode_tmp<0x20 || GKS_(VK_CONTROL)<0 || GKS_(VK_MENU)<0){*/
if(charcode_tmp<0x20 || charcode_tmp>0x7e){
if(cqflag>0 && cqflag%2==0){
/*extraline(1);*/cqflag=0;
if(filerflag) {if(dialogflag>0 && imm_restart_flag==1) imm_restart();}
else imm_restart();
}
else if(cqflag%2==1) cqflag++;

return;
}

if(usflag==1) return;
/*if(driveflag) return;*/

if(menuflag>0){
return;
}/**if(menuflag)*******************************/

if(dialogflag>0){
charcode=charcode_tmp;

if(cqflag==2){
/*overwrite();
insertion_cc_dialog(charcode);
extraline(1);*/cqflag=0;
if(filerflag) {if(imm_restart_flag==1) imm_restart();}
else imm_restart();
}
else if(cqflag==4 || cqflag==6){    /* 4(<- ex. Esc Q) and 6 */
/*if(noelineflag==0) extraline(1);else noelineflag=0;*/
cqflag=0;
if(filerflag) {if(imm_restart_flag==1) imm_restart();}
else imm_restart();
}
else{
overwrite();
insertion_dialog(charcode);
}

if(BitBltflag_==0) {BitBlt_dialog(2);csr();}
else if(BitBltflag_==1)             csr();
else{}
```

```
return;
}/**if(dialogflag)******************************/
}/** WM_func_CHAR **/


void InputPosition(HIMC himc,int icsr,int jcsr)
{
int dx,dy;

myime.dwStyle=CFS_POINT;

if(dialogflag>0) {dx=(icsr+DI_d)*UDX;dy=(jcsr+DJ_d)*UDY;}
else             {dx=(icsr+DI)*UDX;dy=(jcsr+DJ)*UDY;}
point.x=dx;
point.y=dy;

myime.ptCurrentPos=point;
ImmSetCompositionWindow(himc,&myime);
}/** InputPosition **/


void WM_funcIME_CHAR(WPARAM wparam)
{
char flag_,function_old;
long k;
unsigned char db[2];

if(menuflag>0){
return;
}/**if(menuflag)******************************/

if(dialogflag>0){
if(HIBYTE(wparam)){
if(dbflag){
db[0]=HIBYTE(wparam);
db[1]=LOBYTE(wparam);

tailcheck_dialog();

flag_=0;

kmax_dialog+=2;
if(kmax_dialog>ASIZEM-1) {beep(50);kmax_dialog-=2;flag_=1;}
else{
k=firstk_dialog+icsr;
/*memcpy(&p_dialog[k+2],&p_dialog[k],kmax_dialog-2-k+1);*/
memcpy_(&p_dialog[0],k+2,&p_dialog[0],k,kmax_dialog-2-k+1);
/*memcpy(&p_dialog[k],&db[0],2);*/
memcpy_(&p_dialog[0],k,&db[0],0,2);
}

/*page_firstk_dialog(firstk_dialog);*/

if(flag_==0){
csr_right_dialog();
}
else{
dbcount=dbsize;
}
}/**if(dbflag)**/

dbcount+=2;
}/**if(HIBYTE)**/
else{
if(dbflag){
if(insertion_dialog(LOBYTE(wparam))==1) dbcount=dbsize;
if(!compflag) dbsize=1;              /* hankaku space */
}/**if(dbflag)**/

dbcount+=1;
}/**else(HIBYTE)**/

if(dbflag==1 && dbcount>=dbsize){   /* > : notice ! */
dbflag=0;
page_firstk_dialog(firstk_dialog);csr();       /* csr() : for Windows 2000 */

if(compflag){
/*myime.dwStyle=CFS_POINT;
point.x=(icsr+DI_d)*UDX;point.y=(jcsr+DJ_d)*UDY+DSHIFT_2;
myime.ptCurrentPos=point;
ImmSetCompositionWindow(himc,&myime);*/
InputPosition(himc,icsr,jcsr);
}
}

return;
}/**if(dialogflag)*******************************/

}/** WM_funcIME_CHAR **/


void WM_funcIME_STARTCOMPOSITION(void)
```

```
{
/*beep(50);delay_(100);beep(50);delay_(100);*/

if(immflag==0){
compflag=1;

himc=ImmGetContext(hwnd);
ImmGetCompositionFont(himc,&myimefont);
myimefont.lfHeight=UDY;
myimefont.lfWidth=UDX;
strcpy(myimefont.lfFaceName,"MSMINCHO");
ImmSetCompositionFont(himc,&myimefont);
}

/*return 1;*/
}/** WM_funcIME_STARTCOMPOSITION **/


void WM_funcIME_COMPOSITION(LPARAM lparam)
{
static unsigned char dbbuf[ASIZE]={0};

/*beep(500);delay_(100);*/

if(lparam & GCS_RESULTSTR){
if(compflag) dbsize=ImmGetCompositionString(himc,GCS_RESULTSTR,dbbuf,sizeof(dbbuf));
else dbsize=2;                      /* space */
dbflag=1;
dbcount=0;
}
else{
/*myime.dwStyle=CFS_POINT;
if(dialogflag>0) {point.x=(icsr+DI_d)*UDX;point.y=(jcsr+DJ_d)*UDY+DSHIFT_2;}
else {point.x=(icsr+DI)*UDX;point.y=(jcsr+DJ)*UDY+DSHIFT_2;}
myime.ptCurrentPos=point;
ImmSetCompositionWindow(himc,&myime);*/
InputPosition(himc,icsr,jcsr);
}

/*return 1;*/
}/** WM_funcIME_COMPOSITION **/


void WM_funcIME_ENDCOMPOSITION(void)
{
/*beep(50);*/
```

```
if(cqflag) {/*beep(50);*//*extraline(1);*/cqflag=0;imm_restart();}
/*if(puts_mline_flag) {puts_mline_flag=0;extraline(1);}*/

if(compflag==0) {if(immflag!=1) csr();imeendflag=0;}
else imeendflag=1;
compflag=0;
dbflag=0;
/*immflag=0;*/                          /* no problem ? */

ImmReleaseContext(hwnd,himc);
}/** WM_funcIME_ENDCOMPOSITION **/
#else
void WM_func_CHAR(unsigned char charcode_tmp)
{
if(charcode_tmp<0x20 || charcode_tmp>0x7e){
if(cqflag>0 && cqflag%2==0){
/*extraline(1);*/cqflag=0;
if(filerflag) {if(dialogflag>0 && imm_restart_flag==1) imm_restart();}
else imm_restart();
}
else if(cqflag%2==1) cqflag++;

return;
}

if(usflag==1) return;

if(menuflag>0){
return;
}/**if(menuflag)******************************/

if(dialogflag>0){
charcode=charcode_tmp;

if(cqflag==2){
/*overwrite();
insertion_cc_dialog(charcode);
extraline(1);*/cqflag=0;
if(filerflag) {if(imm_restart_flag==1) imm_restart();}
else imm_restart();
}
else if(cqflag==4 || cqflag==6){    /* 4(<- ex. Esc Q) and 6 */
/*if(noelineflag==0) extraline(1);else noelineflag=0;*/
cqflag=0;
if(filerflag) {if(imm_restart_flag==1) imm_restart();}
else imm_restart();
}
else{
overwrite();
insertion_dialog(charcode);
}

return;
}/**if(dialogflag)******************************/
}/** WM_func_CHAR **/


char gettype_sdb(long k)
{
char type;
unsigned char s[2];

s[0]=stock_db[k];
/*s[1]=stock_db[k+1];*/

type=gettype(1,s[0]/*,s[1]*/,k,kmax_sdb);

return type;
}/** gettype_sdb **/


void InputPosition(XIC ic,int icsr,int jcsr)
{
int dx,dy;
POINT point;
XVaNestedList list;

if(style & XIMPreeditPosition){}
else return;

if(dialogflag>0) {dx=(icsr+DI_d)*UDX;dy=(jcsr+DJ_d)*UDY;}
else             {dx=(icsr+DI)*UDX;dy=(jcsr+DJ)*UDY;}
point.x=dx;
point.y=dy+FSIZE;

list=XVaCreateNestedList(0,XNSpotLocation,&point,NULL);
XSetICValues(ic,XNPreeditAttributes,list,NULL);
XFree(list);
}/** InputPosition **/
```

```
void WM_funcIME_CHAR(unsigned char *buf_Xmb)
{
char flag_,function_old,type;
long k,k_sdb;
unsigned char db[2];

strcpy(stock_db,buf_Xmb);
dbsize=strlen(stock_db);
kmax_sdb=dbsize-1;

if(menuflag>0){
return;
}/**if(menuflag)*********************************/

if(dialogflag>0){
dbflag=1;

k_sdb=0;
while(1){
type=gettype_sdb(k_sdb);

if(type==3){
db[0]=stock_db[k_sdb];
db[1]=stock_db[k_sdb+1];

tailcheck_dialog();

flag_=0;

kmax_dialog+=2;
if(kmax_dialog>ASIZEM-1) {beep(50);kmax_dialog-=2;flag_=1;}
else{
k=firstk_dialog+icsr;
memmove(&p_dialog[k+2],&p_dialog[k],kmax_dialog-2-k+1);
memmove(&p_dialog[k],&db[0],2);
}

/*page_firstk_dialog(firstk_dialog);*/

if(flag_==0){
csr_right_dialog();
}
else{
break;
}
k_sdb+=2;
}/**if(type)**/
else{
if(insertion_dialog(stock_db[k_sdb])==1) break;

k_sdb+=1;
}/**else(type)**/

if(k_sdb>kmax_sdb) break;
}/**while(1)**/

dbflag=0;
page_firstk_dialog(firstk_dialog);

InputPosition(ic,icsr,jcsr);

return;
}/**if(dialogflag)******************************/
}/** WM_funcIME_CHAR **/


void initIME(void)
{
int width,height,h;
unsigned char buf[11];

setlocale(LC_ALL,"");
if(XSupportsLocale()==False) exit(1);
if(XSetLocaleModifiers("")==NULL) exit(1);
if((ime=XOpenIM(d,NULL,NULL,NULL))==NULL) exit(1);

style=InputStyle(ime);



/*strcpy(fs1,fs2[fontnum]);*/
if(0) strcpy(fs1,fs2[3]);
else if(0){
/* scalable_ */
h=UDY+/*dh*/0;
h=max(min(h,64),8);
/*printf(" UDX=%d\n",UDX);
printf(" UDY=%d\n",UDY);
printf(" dh=%d\n",dh);
printf(" h=%d\n",h);*/
```

```
if(fontnum==0)
strcpy(fs1,"-*-fixed-medium-r-normal--");
else if(fontnum==1)
strcpy(fs1,"-*-mincho-medium-r-normal--");
else if(fontnum==2)
strcpy(fs1,"-*-gothic-medium-r-normal--");
else
strcpy(fs1,"-*-*-medium-r-normal--");

/*itoa(abs(h),buf,10)*/gcvt(abs(h),3,buf);
strcat(fs1,buf);
strcat(fs1,"-*");
strcat(fs1,"-*-*-*-*-*-*");
}

/*font_fs=XCreateFontSet(d,fs1,&mlist,&mcount,&def);*/



ic=InputContext(ime,style,font_fs,w);
XGetICValues(ic,XNFilterEvents,&mask,NULL);
}/** initIME **/


XIMStyle InputStyle(XIM ime)
{
int i,j,k;
XIMStyles *ime_styles;
static XIMStyle preedit[]={XIMPreeditPosition,XIMPreeditArea,XIMPreeditNothing,0};
static XIMStyle status[]={XIMStatusArea,XIMStatusNothing,0};

XGetIMValues(ime,XNQueryInputStyle,&ime_styles,NULL);

i=0;
while(1){

j=0;
while(1){

for(k=0;k<ime_styles->count_styles;k++){
if((preedit[i] & ime_styles->supported_styles[k]) &&
   (status[j] & ime_styles->supported_styles[k]))
return ime_styles->supported_styles[k];
}

j++;
if(status[j]==0) break;
}

i++;
if(preedit[i]==0) break;
}

return 0;
}/** InputStyle **/


XIC InputContext(XIM ime,XIMStyle style,XFontSet font_fs_auto,Window w)
{
int dx,dy;
XIC ic;
XVaNestedList list;
XPoint point;

dx=(0+DI)*UDX;dy=(0+DJ)*UDY;
point.x=dx;
point.y=dy+FSIZE;

list=XVaCreateNestedList(0,XNFontSet,font_fs_auto,
                           XNSpotLocation,&point,NULL);
ic=XCreateIC(ime,XNInputStyle,style,
                 XNClientWindow,w,
                 XNPreeditAttributes,list,XNStatusAttributes,list,NULL);
XFree(list);
if(ic==NULL) exit(1);

return ic;
}/** InputContext **/
#endif


void csr_to_1(void)
{
int dy=0;
long k;

#if WX==1
XSetFunction(d,gcdisplay,GXxor);
#endif
bitbltflag=1;

if(dialogflag==0){
}/**if(dialogflag)**/
else{
k=firstk_dialog+icsr;
if(ishead_dialog(k)==0){
if(gettype_dialog(k)!=3)
bitblt(-3,0*UDX,0*UDY,UDX,CSRDY,
         (icsr+DI_d)*UDX,(jcsr+DJ_d)*UDY+(UDY-CSRDY)+dy);  /* dialog(single byte) */
else
bitblt(-3,0*UDX,0*UDY,UDX*2,CSRDY,
         (icsr+DI_d)*UDX,(jcsr+DJ_d)*UDY+(UDY-CSRDY)+dy);
}
else
bitblt(-3,0*UDX,0*UDY,UDX*2,CSRDY,
         (icsr-1+DI_d)*UDX,(jcsr+DJ_d)*UDY+(UDY-CSRDY)+dy);
}/**else(dialogflag)**/

#if WX==1
XSetFunction(d,gcdisplay,GXcopy);
#endif
bitbltflag=0;
}/** csr_to_1 **/


void csr(void)
{
int dy=0;
long k;

/*XSetFunction(d,gcdisplay,GXxor);*/
/*bitbltflag=1;*/

/*if(dialogflag==0 && menuflag==0 && filerflag==0){
if(cut>0) indicator(1);
else {if(indicationflag) {indicationflag=0;indicator(0);}}
}
else BitBlt_indicator();*/

csr_to_1();

if(dialogflag==0){
}/**if(dialogflag)**/
else{
k=firstk_dialog+icsr;
if(ishead_dialog(k)==0){
if(gettype_dialog(k)!=3)
bitblt(3,0*UDX,0*UDY,UDX,CSRDY,
         (icsr+DI_d)*UDX,(jcsr+DJ_d)*UDY+(UDY-CSRDY)+dy);  /* dialog(single byte) */
else
bitblt(3,0*UDX,0*UDY,UDX*2,CSRDY,
         (icsr+DI_d)*UDX,(jcsr+DJ_d)*UDY+(UDY-CSRDY)+dy);
}
else
bitblt(3,0*UDX,0*UDY,UDX*2,CSRDY,
         (icsr-1+DI_d)*UDX,(jcsr+DJ_d)*UDY+(UDY-CSRDY)+dy);
}/**else(dialogflag)**/

/*XSetFunction(d,gcdisplay,GXcopy);*/
/*bitbltflag=0;*/

csr_to_1();

BitBltflag=0;
BitBltflag_=0;
}/** csr **/


void memcpy_(unsigned char *p_dst,long k_dst,unsigned char *p_src,long k_src,long sz)
{
long jump,i;

jump=k_dst-k_src;

if(jump>0){  /* up */
for(i=sz-1;i>=0;i--) p_dst[k_dst+i]=p_src[k_src+i];
}
else if(jump<0){  /* down */
for(i=0;i<=sz-1;i++) p_dst[k_dst+i]=p_src[k_src+i];
}
else if(jump==0){  /* parallel */
for(i=0;i<=sz-1;i++) p_dst[k_dst+i]=p_src[k_src+i];
}
}/** memcpy_ **/


#if WX==0
void left_keydowns_dialog(void)
{
char gotoflag;

gotoflag=1;

if(cqflag==6){
```

```
if(GKS('S')<0){
  csr_row_home_dialog();}
else if(GKS('D')<0){
  csr_row_end_dialog();}

else {gotoflag=-1;}
}/**if(cqflag)**/
else {gotoflag=0;}

if(/*cqflag==6*/gotoflag==-1){
  if(GKS(VK_F12)<0 || GKS(VK_PAUSE)<0 || GKS(VK_F1)<0) cqflag=0;  /* Esc+ */
  /*BitBltflag_=2;*/
  goto end_lk_dialog;}

if(gotoflag==1) goto end_lk_dialog;else gotoflag=1;

if(GKS(VK_DELETE)<0){
  deletion_dialog();}
else if(GKS_(VK_SHIFT)>=0 && GKS_(VK_CONTROL)>=0 && GKS(VK_BACK)<0){
  backspace_dialog();}

else if(GKS(VK_UP)<0){
  restore_dialog();}
else if(GKS(VK_DOWN)<0){
  clear_dialog(1);}
else if(GKS_(VK_CONTROL)>=0 && GKS(VK_LEFT)<0){
  csr_left_dialog();}
else if(GKS_(VK_CONTROL)>=0 && GKS(VK_RIGHT)<0){
  csr_right_dialog();}

else if(GKS(VK_HOME)<0 || (GKS_(VK_CONTROL)<0 && GKS(VK_LEFT)<0)){
  csr_row_home_dialog();}
else if(GKS(VK_END)<0 || (GKS_(VK_CONTROL)<0 && GKS(VK_RIGHT)<0)){
  csr_row_end_dialog();}

else if(GKS(VK_PRIOR)<0){
  page_up_dialog();}
else if(GKS(VK_NEXT)<0){
  page_down_dialog();}

else if(GKS_(VK_CONTROL)<0 && GKS(VK_INSERT)<0){
  if(insorover==0) insorover=1;else insorover=0;
  /*extraline(1);*/BitBltflag_=2;}

/*else if(GKS_(VK_SHIFT)<0 && GKS_(VK_CONTROL)<0 && GKS('I')<0){
  cqflag=1;prompt_cq(0);}
```

```
else if(GKS_(VK_CONTROL)<0 && GKS('Q')<0){
  cqflag=5;prompt_cq(2);}*/
else if(GKS(VK_TAB)<0){
  overwrite();
  insertion_dialog(0x09);}

else if(GKS_(VK_CONTROL)<0 && GKS('G')<0){
  deletion_dialog();}
else if(GKS_(VK_SHIFT)>=0 && GKS_(VK_CONTROL)<0 && GKS('H')<0){
  backspace_dialog();}

else if(GKS_(VK_CONTROL)<0 && GKS('E')<0){
  restore_dialog();}
else if(GKS_(VK_CONTROL)<0 && GKS('X')<0){
  clear_dialog(1);}
else if(GKS_(VK_CONTROL)<0 && GKS('S')<0){
  csr_left_dialog();}
else if(GKS_(VK_CONTROL)<0 && GKS('D')<0){
  csr_right_dialog();}

else if(GKS_(VK_CONTROL)<0 && GKS('R')<0){
  page_up_dialog();}
else if(GKS_(VK_CONTROL)<0 && GKS('C')<0){
  page_down_dialog();}

else if(GKS_(VK_CONTROL)<0 && GKS('M')<0){
  trim_dialog();
  /*if(GKS_(VK_SHIFT)<0) use_selector_flag=1;else use_selector_flag=0;*/
  dialogflag=2;refill=0;BitBltflag_=2;}

else {BitBltflag_=2;}

end_lk_dialog:{}
}/** left_keydowns_dialog **/
#else
void left_keydowns_dialog(void)
{
char gotoflag;

gotoflag=1;

if(cqflag==6){
if(GKS('S')<0 || GKS('s')<0){
  csr_row_home_dialog();}
else if(GKS('D')<0 || GKS('d')<0){
  csr_row_end_dialog();}
```

```
else {gotoflag=-1;}
}/**if(cqflag)**/
else {gotoflag=0;}

if(/*cqflag==6*/gotoflag==-1){
  if(GKS(XK_F12)<0 || GKS(XK_Pause)<0 || GKS(XK_F1)<0) cqflag=0;  /* Esc+ */
  /*BitBltflag_=2;*/
  goto end_lk_dialog;}

if(gotoflag==1) goto end_lk_dialog;else gotoflag=1;

if(GKS(XK_Delete)<0){
  deletion_dialog();}
else if(GKS_(ShiftMask)>=0 && GKS_(ControlMask)>=0 && GKS(XK_BackSpace)<0){
  backspace_dialog();}

else if(GKS(XK_Up)<0){
  restore_dialog();}
else if(GKS(XK_Down)<0){
  clear_dialog(1);}
else if(GKS_(ControlMask)>=0 && GKS(XK_Left)<0){
  csr_left_dialog();}
else if(GKS_(ControlMask)>=0 && GKS(XK_Right)<0){
  csr_right_dialog();}

else if(GKS(XK_Home)<0 || (GKS_(ControlMask)<0 && GKS(XK_Left)<0)){
  csr_row_home_dialog();}
else if(GKS(XK_End)<0 || (GKS_(ControlMask)<0 && GKS(XK_Right)<0)){
  csr_row_end_dialog();}

else if(GKS(XK_Prior)<0){
  page_up_dialog();}
else if(GKS(XK_Next)<0){
  page_down_dialog();}

else if(GKS_(ControlMask)<0 && GKS(XK_Insert)<0){
  if(insorover==0) insorover=1;else insorover=0;
  /*extraline(1);*/BitBltflag_=2;}

/*else if(GKS_(ShiftMask)<0 && GKS_(ControlMask)<0 && (GKS('I')<0 || GKS('i')<0)){
  cqflag=1;prompt_cq(0);}
else if(GKS_(ControlMask)<0 && (GKS('Q')<0 || GKS('q')<0)){
  cqflag=5;prompt_cq(2);}*/
else if(GKS(XK_Tab)<0){
  overwrite();
  insertion_dialog(0x09);}

else if(GKS_(ControlMask)<0 && (GKS('G')<0 || GKS('g')<0)){
  deletion_dialog();}
else if(GKS_(ShiftMask)>=0 && GKS_(ControlMask)<0 && (GKS('H')<0 || GKS('h')<0)){
  backspace_dialog();}

else if(GKS_(ControlMask)<0 && (GKS('E')<0 || GKS('e')<0)){
  restore_dialog();}
else if(GKS_(ControlMask)<0 && (GKS('X')<0 || GKS('x')<0)){
  clear_dialog(1);}
else if(GKS_(ControlMask)<0 && (GKS('S')<0 || GKS('s')<0)){
  csr_left_dialog();}
else if(GKS_(ControlMask)<0 && (GKS('D')<0 || GKS('d')<0)){
  csr_right_dialog();}

else if(GKS_(ControlMask)<0 && (GKS('R')<0 || GKS('r')<0)){
  page_up_dialog();}
else if(GKS_(ControlMask)<0 && (GKS('C')<0 || GKS('c')<0)){
  page_down_dialog();}

else if(GKS_(ControlMask)<0 && (GKS('M')<0 || GKS('m')<0)){
  trim_dialog();
  /*if(GKS_(ShiftMask)<0) use_selector_flag=1;else use_selector_flag=0;*/
  dialogflag=2;refill=0;BitBltflag_=2;}

else {BitBltflag_=2;}

end_lk_dialog:{}
}/** left_keydowns_dialog **/
#endif



#if WX==0
void mainroop(void)
{
MSG msg;

while(GetMessage(&msg,NULL,0,0)){
TranslateMessage(&msg);
DispatchMessage(&msg);
if(refill!=1) break;
}/**while(GetMessage)**/
}/** mainroop **/
#else
void mainroop(void)
```

```
{
while(1){
kbhit_();
if(refill!=1) break;
}/**while(1)**/
}/** mainroop **/
#endif


void title(char *str)
{
int i,j,dx,dy,dx_;
int length;

length=strlen(str);

if(dialogflag){
i=DI_d;j=DJ_d-1;dx=i*UDX;dy=j*UDY-1;    /* large */
}
else if(menuflag){
i=1+DI_m;j=0;dx=(i+DI)*UDX;dy=(j+DJ)*UDY;    /* large */
}
else{}

setstccolor(0);

dx_=dx;i=0;
while(1){
dx=dx_+i*UDX;
stc(/*1*/2,dx,dy,&str[i],1);

i++;
if(i==length) break;
}
}/** title **/


void before_mainroop(char *str)
{
int i,j,dx,dy;
int length;

icsr=0;jcsr=0;

i=DI_d-1;j=DJ_d-1;dx=i*UDX;dy=j*UDY;    /* large */
paint(/*1*/2,dx,dy,UDX*((CD+1)+2),UDY*(1+2),7);
```

```
title(str);                         /* title */

length=strlen(p_dialog);
if(length<ASIZEM){
p_dialog[length]=0x1a;
p_dialog[length+1]='\0';
}
else{}                              /* impossible */

strcpy(p_restore,p_dialog);

if(/*!driveflag*/1){
if(noclearflag==0) clear_dialog(0);
else{                               /* in dlgproc_SAVE() */
kmax_dialog=strlen(p_dialog)-1;
text_end_dialog();}

csr();
}
}/** before_mainroop **/


void after_mainroop(void)
{
refill=1;
}/** after_mainroop **/


void csr_tab_dialog(char leftorright)
{
char flag_tab,flag_cc,flag_2b,type;
int icsr_,ris;
long k;

k=firstk_dialog;
icsr_=0;
flag_tab=0;
flag_cc=0;
flag_2b=0;

if(icsr==0){
}/**if(icsr)**/
else{
while(1){
type=gettype_dialog(k);
if(type<=2){
k++;

if(type<=0){
icsr_++;
if(icsr_==icsr) {/*flag=0;*/break;}
}/**if(type)**/
else if(type==1){                    /* Tab */
icsr_++;
if(icsr_==icsr) {flag_tab=0;break;}
if(icsr_>icsr) {flag_tab=1;break;}
}/**else if(type)**/
else{                                /* Control code */
icsr_++;
if(icsr_==icsr) {flag_cc=0;break;}
if(icsr_>icsr) {flag_cc=1;break;}
}/**else(type)**/
}/**if(type)**/
else if(type==3){
k+=2;

icsr_+=2;
if(icsr_==icsr) {flag_2b=0;break;}
if(icsr_>icsr) {flag_2b=1;break;}
}/**else if(type)**/
else{
}/**else(type)**/
}/**while(1)**/

if(leftorright==0)
icsr=icsr_-flag_2b*2;
else{
icsr=icsr_;
}
}/**else(icsr)**/
}/** csr_tab_dialog **/


char gettype_dialog(long k)
{
char type;
unsigned char s[2];

s[0]=p_dialog[k];
/*s[1]=p_dialog[k+1];*/
```

```
type=gettype(1,s[0]/*,s[1]*/,k,kmax_dialog);

return type;
}/** gettype_dialog **/


long gethead_dialog(char flag,long member)
{
char type;
long k,dk_auto;

k=0;dk_auto=0;

while(1){
if(k==member) return k;
if(k>member){
if(flag==0) k-=dk_auto;else k=k;
return k;
}

type=gettype_dialog(k);

if(type<=2) {dk_auto=1;k+=dk_auto;}
else if(type==3) {dk_auto=2;k+=dk_auto;}
else{}
}
}/** gethead_dialog **/


int ishead_dialog(long member)
{
char type;
int dk_auto;
long k;

k=firstk_dialog;dk_auto=0;

while(1){
if(k==member) return 0;
if(k>member) return dk_auto;

type=gettype_dialog(k);

if(type<=2) {dk_auto=1;k+=dk_auto;}
else if(type==3) {dk_auto=2;k+=dk_auto;}
else{}
}
}/** ishead_dialog **/


void backspace_dialog(void)
{
char flag;
long k,k_icsr,firstk_dialog_;

/*if(driveflag) return;*/

flag=csr_left_dialog();

if(flag!=1){
lumpflag_dialog=1;
deletion_dialog();
lumpflag_dialog=0;

if(flag==2){                        /* scrolled up */
k_icsr=firstk_dialog+icsr;

if(firstk_dialog-(CD-1)>0){
firstk_dialog_=max(min(firstk_dialog-(CD-1),kmax_dialog),0);  /* protection */
firstk_dialog=gethead_dialog(1,firstk_dialog_);    /* for jp */
page_firstk_dialog(firstk_dialog);
icsr=k_icsr-firstk_dialog;
}
else{
page_firstk_dialog(0);
icsr=k_icsr-0;
}
}/**if(flag)**/
else{
page_firstk_dialog(firstk_dialog);
}/**else(flag)**/
}/**if(flag!)**/
}/** backspace_dialog **/


int insertion_dialog(unsigned char charcode)
{
char flag_;
long k;

/*if(driveflag) return 1;*/
```

```
tailcheck_dialog();

flag_=0;

kmax_dialog++;
if(kmax_dialog>ASIZEM-1) {beep(50);kmax_dialog--;flag_=1;}
else{
k=firstk_dialog+icsr;
/*memcpy(&p_dialog[k+1],&p_dialog[k],kmax_dialog-1-k+1);*/
memcpy_(&p_dialog[0],k+1,&p_dialog[0],k,kmax_dialog-1-k+1);
p_dialog[k]=charcode;
}

page_firstk_dialog(firstk_dialog);

if(flag_==0){
csr_right_dialog();
/*if(puts_mline_flag) {puts_mline_flag=0;extraline(1);}*/
return 0;
}
else{
return 1;
}
}/** insertion_dialog **/


int deletion_dialog(void)
{
char type;
long k,dk;

/*if(driveflag) return 1;*/

tailcheck_dialog();

k=firstk_dialog+icsr;
if(k==kmax_dialog) return 1;

type=gettype_dialog(k);

if(type<=2){
dk=1;
/*memcpy(&p_dialog[k],&p_dialog[k+dk],kmax_dialog-(k+dk)+1);*/
memcpy_(&p_dialog[0],k,&p_dialog[0],k+dk,kmax_dialog-(k+dk)+1);
kmax_dialog-=dk;
```

```
}/**if(type)**/
else if(type==3){
dk=2;
/*memcpy(&p_dialog[k],&p_dialog[k+dk],kmax_dialog-(k+dk)+1);*/
memcpy_(&p_dialog[0],k,&p_dialog[0],k+dk,kmax_dialog-(k+dk)+1);
kmax_dialog-=dk;
}/**else if(type)**/
else{
}/**else(type)**/

if(overwriteflag==1) return 1;

page_firstk_dialog(firstk_dialog);
/*if(puts_mline_flag) {puts_mline_flag=0;extraline(1);}*/

return 0;
}/** deletion_dialog **/


void while_puts_show_dialog(long k)
{
char TextOutflag,type,hdc=2;
int i,j,dx,dy;
unsigned char s[1],s_[1];
unsigned char jis[2];

TextOutflag=1;
i=0;j=0;

while(1){
s[0]=p_dialog[k];
type=gettype_dialog(k);

if(type<=2){
if(TextOutflag){
if(s[0]>=0x20 && type==0)
setstccolor(bfset[WB].fore);
else if(type==-1)
setstccolor(/*12*/bfset[WB].fore);
/*else if(s[0]==0x1a)*/
else if(k==kmax_dialog)
setstccolor(12);
/*else if(s[0]=='\n')
setstccolor(RETURN);*/
/*else if(s[0]==0x09)
setstccolor(TABCOLOR);*/
```

```
else
setstccolor(/*CC*/RTC);

dx=(i+DI_d)*UDX;dy=(j+DJ_d)*UDY-1+WX*(1);

if(s[0]>=0x20 && type==0)
stc(hdc,dx,dy,s,1);
/*else if(s[0]=='\n'){
s_[0]=0x0d;
stc(hdc,dx,dy,s_,1);
}*/
/*else if(s[0]==0x1a){*/
else if(k==kmax_dialog){
s_[0]=/*0x0d*/dummy_R;
stc(hdc,dx,dy,s_,1);
}
else if(type==-1){
s_[0]=0x0d;
stc(hdc,dx,dy,s_,1);
}
/*else if(s[0]==0x09){
s_[0]=0x0d;
stc(hdc,dx,dy,s_,1);
}*/
else{
if(s[0]==0x7f) s[0]=0x00;
s_[0]=cc[s[0]];
stc(hdc,dx,dy,s_,1);
}
}/**if(TextOutflag)**/

/*if(k==kmax_dialog) break;*/

k++;
i++;
if(i==CD) break;
}/**if(type)**/
else if(type==3){
if(TextOutflag){
jis[0]=p_dialog[k];
jis[1]=p_dialog[k+1];

dx=(i+DI_d)*UDX;dy=(j+DJ_d)*UDY;
setstccolor(bfset[WB].fore);
stc(hdc,dx,dy,jis,2);
}/**if(TextOutflag)**/
```

```
/*if(k==kmax_dialog) break;*/              /* ? */

k+=2;
i+=2;
if(i>=CD) break;
}/**else if(type)**/
else{
}/**else(type)**/

if(k>kmax_dialog) break;           /* new break */
}
}/** while_puts_show_dialog **/


void text_end_dialog(void)
{
long firstk_dialog_;

firstk_dialog_=max(/*min(*/kmax_dialog-(CD-1)/*,kmax_dialog)*/,0);  /* protection */
firstk_dialog=gethead_dialog(1,firstk_dialog_);    /* for jp */

page_firstk_dialog(firstk_dialog);
csr_row_end_dialog();
}/** text_end_dialog **/


void page_down_dialog(void)
{
long firstk_dialog_;

if(firstk_dialog+CD-1+icsr<=kmax_dialog)
firstk_dialog_=max(min(firstk_dialog+CD-1,kmax_dialog),0);  /* protection */
else
firstk_dialog_=max(/*min(*/kmax_dialog-icsr/*,kmax_dialog)*/,0);  /* protection */
firstk_dialog=gethead_dialog(1,firstk_dialog_);    /* for jp */

page_firstk_dialog(firstk_dialog);
/*within_linemax_dialog();*/
tailcheck_dialog();
}/** page_down_dialog **/


void page_up_dialog(void)
{
long firstk_dialog_;
```

```
firstk_dialog_=max(min(firstk_dialog-(CD-1),kmax_dialog),0);  /* protection */
firstk_dialog=gethead_dialog(1,firstk_dialog_);    /* for jp */

page_firstk_dialog(firstk_dialog);
/*within_linemax_dialog();*/
tailcheck_dialog();
}/** page_up_dialog **/


void trim_dialog(void)
{
p_dialog[kmax_dialog]='\0';
}/** trim_dialog **/


void restore_dialog(void)
{
/*if(driveflag) return;*/

strcpy(p_dialog,p_restore);

kmax_dialog=strlen(p_dialog)-1;
text_end_dialog();

/*if(puts_mline_flag) {puts_mline_flag=0;extraline(1);}*/
}/** restore_dialog **/


void clear_dialog(char flag)
{
/*if(driveflag) return;*/

kmax_dialog=0;
p_dialog[0]=0x1a;
p_dialog[1]='\0';

/*within_linemax_dialog();*/
tailcheck_dialog();
page_firstk_dialog(firstk_dialog);

/*if(flag) {if(puts_mline_flag) {puts_mline_flag=0;extraline(1);}}*/
}/** clear_dialog **/


void page_firstk_dialog(long k)
```

```
{
int i,j,dx,dy;

firstk_dialog=max(min(k,kmax_dialog),0);   /* protection */

if(lumpflag_dialog==1) return;
if(dbflag==1) return;

i=DI_d;j=DJ_d;dx=i*UDX;dy=j*UDY;    /* small */
cleardevice_(/*1*/2,dx,dy,UDX*(CD+1),UDY);
while_puts_show_dialog(firstk_dialog);

BitBlt_dialog(2);
}/** page_firstk_dialog **/


void BitBlt_dialog(char hdc)
{
int i,j,dx,dy;

i=DI_d-1;j=DJ_d-1;dx=i*UDX;dy=j*UDY;    /* large */
bitblt(hdc,dx,dy,UDX*((CD+1)+2),UDY*(1+2),dx,dy);

BitBltflag_=1;
}/** BitBlt_dialog **/


void tailcheck_dialog(void)
{
within_linemax_dialog();

csr_tab_dialog(0);
}/** tailcheck_dialog **/


void within_linemax_dialog(void)
{
/*if(firstk_dialog>kmax_dialog) firstk_dialog=kmax_dialog;*/
firstk_dialog=max(min(firstk_dialog,kmax_dialog),0);   /* protection */

if(firstk_dialog+icsr>kmax_dialog) icsr=kmax_dialog-firstk_dialog;
}/** within_linemax_dialog **/


void csr_row_home_dialog(void)
{
```

```
icsr=0;
}/** csr_row_home_dialog **/


void csr_row_end_dialog(void)
{
icsr=CD-1;

/*within_linemax_dialog();*/
tailcheck_dialog();
}/** csr_row_end_dialog **/


char csr_left_dialog(void)
{
icsr--;
within_linemax_dialog();

if(icsr<0){
icsr=0;

if(scroll_up_dialog()==1)
return 1;
else{
/*csr_tab_dialog(0);*/
return 2;}
}

csr_tab_dialog(0);

return 0;
}/** csr_left_dialog **/


void csr_right_dialog(void)
{
char type;
long k,k_icsr;

within_linemax_dialog();
k=firstk_dialog+icsr;
if(k==kmax_dialog) return;

icsr++;
csr_tab_dialog(1);
k_icsr=firstk_dialog+icsr;
```

```
while(1){
if(icsr>CD-1){
scroll_down_dialog();
icsr=k_icsr-firstk_dialog;}
else break;
}
}/** csr_right_dialog **/


int scroll_down_dialog(void)
{
if(firstk_dialog>=kmax_dialog) return 1;

firstk_dialog++;
firstk_dialog=gethead_dialog(1,firstk_dialog);    /* for jp */
page_firstk_dialog(firstk_dialog);

/*within_linemax_dialog();*/
tailcheck_dialog();

return 0;
}/** scroll_down_dialog **/


int scroll_up_dialog(void)
{
if(firstk_dialog<1) return 1;

firstk_dialog--;
firstk_dialog=gethead_dialog(0,firstk_dialog);    /* for jp */
page_firstk_dialog(firstk_dialog);

/*within_linemax_dialog();*/
tailcheck_dialog();

return 0;
}/** scroll_up_dialog **/

/************************** <- dialog functions ***************************/

void fsave(char *str)
{
int i,j;
static char fname[ASIZE]="test.bin";
```

```
dialogflag=1;

if(strlen(str)==0){
begin:
strcpy(p_dialog,fname);
before_mainroop("Save");
mainroop();                         /* p_dialog */
after_mainroop();
if(dialogflag==3) goto end;
strcpy(fname,p_dialog);

if((fp=fopen(fname,"w+b"))==NULL){
printf(" Reinput a filename\n");
strcpy(fname,"");goto begin;}
}
else{
fp=fopen(str,"w+b");
}

/* not used */
xy.xt1=3*(RESO-1);
xy.xt2=-9*(RESO-1);
xy.yt=4*(RESO-1);
xy.CPM=CPMAX;
xy.RCM=RCMAX;

fwrite(&xy,12,1,fp);
for(j=0;j<yt+2;j++)
for(i=0;i<xt+2;i++)
fwrite(&pixel[i][j],1,1,fp);

fclose(fp);
printf(" Save\n");

end:
BitBlt_dialog(1);
dialogflag=0;
}/** fsave **/


void fload(char *str)
{
char val;
int old,i,j;
static char fname[ASIZE]="test.bin";
```

```
dialogflag=1;

if(strlen(str)==0){
begin:
strcpy(p_dialog,fname);
before_mainroop("Load");
mainroop();                           /* p_dialog */
after_mainroop();
if(dialogflag==3) goto end;
strcpy(fname,p_dialog);

if((fp=fopen(fname,"r+b"))==NULL){
printf(" Reinput a filename\n");
strcpy(fname,"");goto begin;}
}
else{
fp=fopen(str,"r+b");
}

fread(&xy,12,1,fp);
/*printf(" %d %d %d\n",xy.xt1,xy.xt2,xy.yt);*/

old=EDGE;EDGE=1;
predraw=0;

/* 16 */
for(j=0;j<yt+2;j++)
for(i=0;i<xt+2;i++){
fread(&val,1,1,fp);
pixel[i][j]=val;
if(val==16) {/*xg_=-1;*/putpixel(i,j,val);}
}

/* -1 */
/*fseek(fp,12,0);
d_trans=1;
;*/

d_trans=0;
fseek(fp,12,0);
EDGE=0;

/* 0~15 */
for(j=0;j<yt+2;j++)
for(i=0;i<xt+2;i++){
fread(&val,1,1,fp);
pixel[i][j]=val;
if(val!=16 && val!=wall) {/*xg_=-1;*/putpixel(i,j,val);}
}

predraw=1;
BitBlt_full();
EDGE=old;
fclose(fp);
printf(" Load\n");

end:
BitBlt_dialog(1);
dialogflag=0;
}/** fload **/


void restore_edge(void)
{
char val;
int old,i,j;

old=EDGE;EDGE=1;
predraw=0;

/* -1 */
/*d_trans=1;
;*/

d_trans=0;
EDGE=0;

/* 0~15 */
for(j=0;j<yt+2;j++)
for(i=0;i<xt+2;i++){
val=pixel[i][j];
if(val!=16 && val!=wall) {/*xg_=id[i][j];*/putpixel(i,j,val);}
}

predraw=1;
BitBlt_full();
EDGE=old;
}/** restore_edge **/


int putpixel(int cx,int cy,int pcolor)
{
```

```
char flag_id;
int k,dx[2],dy[2],pencolor,num;
POINT vertex[7];

if(cx<0 || cy<0) return 1;
if(!GRPH) goto end;
if(c_trans==1) goto end;
if(c_trans==2 && searchflag==0) goto end;

#if HEX_or_SQR==0
dx[0]=get_dx_h(cx,cy);
dy[0]=get_dy_h(cx,cy);

vertex[0].x=dx[0];               vertex[0].y=dy[0];
vertex[2].x=get_dx_h(cx+1,cy+0);vertex[2].y=get_dy_h(cx+1,cy+0);
vertex[4].x=get_dx_h(cx+1,cy+1);vertex[4].y=get_dy_h(cx+1,cy+1);

vertex[1].x=vertex[4].x      ;vertex[1].y=vertex[0].y-dy_hex;
vertex[3].x=vertex[2].x      ;vertex[3].y=vertex[4].y-dy_hex;
vertex[5].x=vertex[0].x      ;vertex[5].y=vertex[4].y-dy_hex;

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

num=6;
#else
dx[0]=X0;
dy[0]=Y0;
if(EDGE==0) {dx[1]=-1;dy[1]=-1;}
else        {dx[1]=0; dy[1]=0;}

vertex[0].x=dx[0]+cx*PIXSIZE;          vertex[0].y=dy[0]+cy*PIXSIZE;
vertex[1].x=dx[0]+(cx+1)*PIXSIZE+dx[1];vertex[1].y=dy[0]+cy*PIXSIZE;
vertex[2].x=dx[0]+(cx+1)*PIXSIZE+dx[1];vertex[2].y=dy[0]+(cy+1)*PIXSIZE+dy[1];
vertex[3].x=dx[0]+cx*PIXSIZE;          vertex[3].y=dy[0]+(cy+1)*PIXSIZE+dy[1];
vertex[4].x=vertex[0].x;               vertex[4].y=vertex[0].y;

num=4;
#endif

if(EDGE==1){
     if(pcolor==15 || pcolor==16){
            pencolor=10;
}
else if(pcolor==0 || pcolor==-1){
if(d_trans) pencolor=bfset[WB].fore;
else        pencolor=15;
}
else        pencolor=15;
}
else{
pencolor=pcolor;
}

if(fieldflag==0 && c_trans==0) {xi=cx;yi=cy;}
/*else                           {xi=-1;yi=-1;}*/



if(EDGE==1){
     if(pcolor==15 || pcolor==16){
            k=15;
}
else if(pcolor==0 || pcolor==-1){
if(d_trans) k=bfset[WB].back;
else        k=0;
}
else        k=pcolor;
Polygon_(vertex,num,k);
Polyline_(vertex,num,pencolor);
}
else if(c_trans==3 && Fill>/*-1*/0){
Polygon_(vertex,num,0/*bdrcolor*/);
Polyline_(vertex,num,8);
putperiod=1;
if(xg==yg)
Polyline_(vertex,num,13);
else
Polyline_(vertex,num,14);
putperiod=0;
}
else if(!zeroFill){
Polygon_(vertex,num,pcolor);
Polyline_(vertex,num,pcolor);
}

end:
if(c_trans==0){
if(cx<=xt && cy<=yt) pixel[cx][cy]=pcolor;
}
if(c_trans==3){
if(Fill==-1 && cx<=xt && cy<=yt) pixel[cx][cy]=pcolor;
```

```
}
else if(c_trans==1){
if(cx<=xt && cy<=yt) pixel_[cx][cy]=pcolor;
}
else if(c_trans==2){
if(cx<=xt && cy<=yt) {pixel_[cx][cy]=pcolor;pixel[cx][cy]=pcolor;}
}

#if YOUR_ART==1
if(fieldflag==0){
if(c_trans==0){
k=0;
while(1){
if(mbr[rcount[CPMAX_-1]].x[ig][k]==-1){
mbr[rcount[CPMAX_-1]].x[ig][k]=cx;
mbr[rcount[CPMAX_-1]].y[ig][k]=cy;
/*if(k>d_trans) d_trans=k;*/
break;
}
else{
k++;
if(k>9) {printf(" k?\n");refill=0;break;}
}
}/**while(1)**/
}/**if(c_trans)**/
else{
}/**else(c_trans)**/
}/**if(fieldflag)**/
else{
}/**else(fieldflag)**/
#endif

return 0;
}/** putpixel **/


int putpixel_noarray(int cx,int cy,int pcolor)
{
int k,dx[2],dy[2],pencolor,num;
POINT vertex[7];

if(HEX_or_SQR==1){
dx[0]=X0;
dy[0]=Y0;
if(EDGE==0) {dx[1]=-1;dy[1]=-1;}
else        {dx[1]=0; dy[1]=0;}
```

```
vertex[0].x=dx[0]+cx*PIXSIZE;          vertex[0].y=dy[0]+cy*PIXSIZE;
vertex[1].x=dx[0]+(cx+1)*PIXSIZE+dx[1];vertex[1].y=dy[0]+cy*PIXSIZE;
vertex[2].x=dx[0]+(cx+1)*PIXSIZE+dx[1];vertex[2].y=dy[0]+(cy+1)*PIXSIZE+dy[1];
vertex[3].x=dx[0]+cx*PIXSIZE;          vertex[3].y=dy[0]+(cy+1)*PIXSIZE+dy[1];
vertex[4].x=vertex[0].x;               vertex[4].y=vertex[0].y;

num=4;
}
else{
dx[0]=get_dx_h(cx,cy);
dy[0]=get_dy_h(cx,cy);

vertex[0].x=dx[0];               vertex[0].y=dy[0];
vertex[2].x=get_dx_h(cx+1,cy+0);vertex[2].y=get_dy_h(cx+1,cy+0);
vertex[4].x=get_dx_h(cx+1,cy+1);vertex[4].y=get_dy_h(cx+1,cy+1);

vertex[1].x=vertex[4].x      ;vertex[1].y=vertex[0].y-dy_hex;
vertex[3].x=vertex[2].x      ;vertex[3].y=vertex[4].y-dy_hex;
vertex[5].x=vertex[0].x      ;vertex[5].y=vertex[4].y-dy_hex;

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

/*for(i=0;i<7;i++) vertex[i].x-=N1*PIXSIZE*1+15;*/

num=6;
}

if(EDGE==1){
     if(pcolor==15 || pcolor==16){
            pencolor=10;
}
else if(pcolor==0 || pcolor==-1){
if(d_trans) pencolor=bfset[WB].fore;
else        pencolor=15;
}
else        pencolor=15;
}
else{
pencolor=pcolor;
}

if(EDGE==1){
     if(pcolor==15 || pcolor==16){
            k=15;
}
else if(pcolor==0 || pcolor==-1){
if(d_trans) k=bfset[WB].back;
else        k=0;
}
else        k=pcolor;
Polygon_(vertex,num,k);
Polyline_(vertex,num,pencolor);
}
else if(c_trans==3 && Fill>/*-1*/0){
Polygon_(vertex,num,0);
Polyline_(vertex,num,8);
putperiod=1;
Polyline_(vertex,num,13);
putperiod=0;
}
else if(1){
Polygon_(vertex,num,pcolor);
Polyline_(vertex,num,pcolor);
}

return 0;
}/** putpixel_noarray **/


void check_rcount(void)
{
int i;
long val[2];

if(GRPH>0){
for(i=0;i<CPMAX_;i++)
printf(" %ld %ld\n",cnt,rcount[i]);
}
else if(1){
if(/*Odd>0 || CPMAX_==5*/1){
val[0]=rcount[0];
for(i=1;i<CPMAX_;i++){
if(rcount[i]!=val[0]) {beep(1000);refill=0;break;}
}

if(refill==0)
printf(" %ld %ld %d:%ld\n",cnt,val[0],i,rcount[i]);
else
printf(" %ld %ld\n",cnt,val[0]);
}/**if(_6dRow, CPMAX_)**/
```

```
else{
val[0]=rcount[0];
for(i=1;i<CPHALF;i++){
if(rcount[i]!=val[0]) {beep(1000);refill=0;break;}
}

if(refill){
val[1]=rcount[CPHALF];
for(i=CPHALF+1;i<CPMAX_;i++){
if(rcount[i]!=val[1]) {beep(1000);refill=0;break;}
}

if(refill)
printf(" %ld %ld %ld\n",cnt,val[0],val[1]);
else
printf(" %ld 1st:%ld 2nd:%ld %d:%ld\n",cnt,val[0],val[1],i,rcount[i]);
}/**if(refill)**/
else{
printf(" %ld 1st:%ld %d:%ld\n",cnt,val[0],i,rcount[i]);
}/**else(refill)**/
}/**else(_6dRow, CPMAX_)**/
}
else{
printf(" %ld %ld %ld\n",cnt,rcount[0],rcount[1]);
if(rcount[0]!=rcount[1]) {beep(100);refill=0;}
}

if(GRPH==0 && cnt==GRPH_0_MAX) {beep(100);refill=0;}
}/** check_rcount **/


void arrayreset(char flag,int wcolor)
{
int i,j,k;

if(flag<=1){                        /* 0, 1 */
i=0;
while(1){

j=0;
while(1){
pixel[i][j]=/*wcolor*/wall;
j++;
if(j==(yt+1)+1) break;
}
i++;
if(i==(xt+1)+1) break;
}
}/**if(flag)**/

if(flag==0){                        /* 0 */
for(i=0;i<Zxd2;i++)
for(j=0;j<Zyd2;j++){
dsp[i][j].cx=-2;dsp[i][j].cy=-2;
}
}/**if(flag)**/

if(flag!=1){                        /* 0, 2 */
for(i=0;i<RCMAX;i++)  /* rcount[CPMAX-1] */
for(j=0;j<CPMAX_;j++)  /* ig */
for(k=0;k<10;k++){    /* putpixel() + pp_() */
mbr[i].x[j][k]=-1;
mbr[i].y[j][k]=-1;
}

/*for(i=0;i<Zxd2;i++)
for(j=0;j<Zyd2;j++)
id[i][j]=-2;*/
}/**if(flag)**/
}/** arrayreset **/


void arrayreset_(void)
{
int i,j;

i=0;
while(1){

j=0;
while(1){
pixel_[i][j]=/*0*//*-2*/wall;
j++;
if(j==(yt+1)+1) break;
}

i++;
if(i==(xt+1)+1) break;
}
}/** arrayreset_ **/
```

```
#if HEX_or_SQR==0
void putdelta(int n,int lr,int x,int y,int color)
{
int i,j;

if(lr==0){
for(j=0;j<=n-1;j++)
for(i=0;i<=j;i++){
if(tmp0!=0) putpixel_(x+i,y+j,color);
else{
if(pixel[x+i][y+j]!=16) putpixel_(x+i,y+j,/*fcolor*/16);
}
kbhit_();if(refill==0) break;
}
}
else{
for(j=0;j<=n-1;j++)
for(i=0;i<=j;i++){
if(tmp0!=0) putpixel_(x+(n-1)-i,y+(n-1)-j,color);
else{
if(pixel[x+(n-1)-i][y+(n-1)-j]!=16) putpixel_(x+(n-1)-i,y+(n-1)-j,/*fcolor*/16);
}
kbhit_();if(refill==0) break;
}
}
}/** putdelta **/


void hexagon(int nx,int ny,int fcolor)
{
int n=Lhex;

nx++;ny++;

putdelta(n,0,nx+0,ny+0,fcolor);
putdelta(n,1,nx+0,ny+0,fcolor);
putdelta(n,0,nx+1*(n-1),ny+0,fcolor);
putdelta(n,1,nx+0,ny+1*(n-1),fcolor);
putdelta(n,0,nx+1*(n-1),ny+1*(n-1),fcolor);
putdelta(n,1,nx+1*(n-1),ny+1*(n-1),fcolor);
}/** hexagon **/
#endif


void field(int wcolor)
```

```
{
int i,j,x,y,n,fcolor=16,dlt,old,old_;
static int cnt_=0;

#if HEX_or_SQR==0
n=Lhex;
#endif

fieldflag=1;
old=EDGE;EDGE=1;
old_=predraw;predraw=0;

#if HEX_or_SQR==0
if(wcolor==-1) tmp0=0;else tmp0=9;

hexagon(/*5*/n-1+1,0,fcolor);
hexagon(0,/*4*/n-1,fcolor);
hexagon(/*14*/(n-1)*3+2,/*5*/n-1+1,fcolor);
hexagon(/*9*/(n-1)*2+1,/*9*/(n-1)*2+1,fcolor);
hexagon(/*4*/n-1,/*13*/(n-1)*3+1,fcolor);
hexagon(/*18*/(n-1)*4+2,/*14*/(n-1)*3+2,fcolor);
hexagon(/*13*/(n-1)*3+1,/*18*/(n-1)*4+2,fcolor);

/* set_DP() */
putpixel_(XDP,YDP,0);                                /* 0 */
putpixel_(XDP-((Lhex-1)*2+1),YDP-((Lhex-1)*1+1),0);  /* 1 */
putpixel_(XDP-(Lhex-1),YDP-((Lhex-1)*2+1),0);        /* 2 */
putpixel_(XDP+Lhex,YDP-(Lhex-1),0);                  /* 3 */
putpixel_(XDP+((Lhex-1)*2+1),YDP+Lhex,0);            /* 4 */
putpixel_(XDP+(Lhex-1),YDP+((Lhex-1)*2+1),0);        /* 5 */
putpixel_(XDP-Lhex,YDP+(Lhex-1),0);                  /* 6 */
#else
for(y=0;y<yt+1;++y)
for(x=0;x<xt+1;++x){
if(wcolor!=-1) putpixel/*_*/(x,y,fcolor);
else {if(pixel[x][y]!=16) putpixel/*_*/(x,y,fcolor);}
kbhit_();if(refill==0) break;
}
#endif

if(YOUR_ART==1 && !cnt_){
dlt=SQSZ;

for(i=0;i<16;i++)
rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,i);
i=16;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=17;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);

i=19;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=20;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=21;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=22;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=23;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
setstccolor(0);
i=16;stc(1,i*dlt+9,(int)(0.27*dlt-WX*1),"R",1);
i=17;stc(1,i*dlt+6,(int)(0.27*dlt-WX*1),"PP",2);

/*i=19;stc(1,i*dlt+6,(int)(0.27*dlt-WX*1),"ID",2);*/
i=20;stc(1,i*dlt+9,(int)(0.27*dlt-WX*1),"C",1);
i=21;stc(1,i*dlt+6,(int)(0.27*dlt-WX*1),"AC",2);
i=22;stc(1,i*dlt+9,(int)(0.27*dlt-WX*1),"S",1);
i=23;stc(1,i*dlt+9,(int)(0.27*dlt-WX*1),"L",1);
setstccolor(bfset[WB].fore);

i=25;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=26;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=27;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=28;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
i=29;rectangle_(1,i*dlt,0-PPDY,(i+1)*dlt,dlt-PPDY,bfset[WB].fore,7);
setstccolor(0);
i=25;stc(1,i*dlt+9,(int)(0.27*dlt-WX*1),"F",1);
i=26;stc(1,i*dlt+9,(int)(0.27*dlt-WX*1),"C",1);
i=27;stc(1,i*dlt+6,(int)(0.27*dlt-WX*1),"AC",2);
i=28;stc(1,i*dlt+9,(int)(0.27*dlt-WX*1),"R",1);
i=29;stc(1,i*dlt+6,(int)(0.27*dlt-WX*1),"48",2);
setstccolor(bfset[WB].fore);

i=9;rectangle_(0,i*dlt,dlt-PPDY+2,(i+1)*dlt,dlt-PPDY+3,bfset[WB].fore,i);
/*printf_("id_1st=",id_1st,DX_,0);*/

cnt_++;
}

predraw=old_;
BitBlt_full();
EDGE=old;
fieldflag=0;
}/** field **/


void rectangle_(char flag,int x1,int y1,int x2,int y2,int color1,int color2)
```

```
{
int i,j;

line(x1,y1,x2,y1,color1);
line(x2,y1,x2,y2,color1);
line(x2,y2,x1,y2,color1);
line(x1,y2,x1,y1,color1);

if(flag)
for(j=y1+1;j<y2;j++)
for(i=x1+1;i<x2;i++)
ppixel(i,j,color2);
}/** rectangle_ **/


void putpixel_(int nx,int ny,int pcolor)
{
putpixel(nx,ny,pcolor);
}/** putpixel_ **/


int getpixel_(int x,int y)
{
#if HEX_or_SQR==0
/* protection */
if(y==Lhex-1 &&
   x>=XDP-(Lhex-1)-((Lhex-1)*2+1) &&
   x<=XDP-(Lhex-1)-((Lhex-1)*2+1)+(Lhex-1)){  /* d__ 1->0->5 */  /* 1 */
x+=Lhex-1+((Lhex-1)*2+1)+(Lhex-1);
y+=2*Lhex-1+((Lhex-1)*1+1)+((Lhex-1)*2+1);
}
else if(x==0 &&
        y<=YDP-1-((Lhex-1)*1+1) &&
        y>=YDP-1-((Lhex-1)*1+1)-(Lhex-1)){  /* b__ 1->0->3 */
x+=2*Lhex-1+((Lhex-1)*2+1)+Lhex;
y+=Lhex+((Lhex-1)*1+1)-(Lhex-1);
}
else if(y==x+2*Lhex-1 &&
        x<=XDP-1-((Lhex-1)*2+1) &&
        x>=XDP-1-((Lhex-1)*2+1)-(Lhex-1)){ /* e__ 1->0->4 */
x+=-(-Lhex)+((Lhex-1)*2+1)+((Lhex-1)*2+1);
y+=-(Lhex-1)+((Lhex-1)*1+1)+Lhex;
}

else if(y==0 &&
        x>=XDP-(Lhex-1)-(Lhex-1) &&
        x<=XDP-(Lhex-1)-(Lhex-1)+(Lhex-1)){  /* d__ 2->0->4 */  /* 2 */
x+=Lhex-1+(Lhex-1)+((Lhex-1)*2+1);
y+=2*Lhex-1+((Lhex-1)*2+1)+Lhex;
}
else if(x==Lhex &&
        y<=YDP-1-((Lhex-1)*2+1) &&
        y>=YDP-1-((Lhex-1)*2+1)-(Lhex-1)){  /* b__ 2->0->5 */
x+=2*Lhex-1+(Lhex-1)+(Lhex-1);
y+=Lhex+((Lhex-1)*2+1)+((Lhex-1)*2+1);
}
else if(y==x-2*Lhex &&
        x>=XDP+1-(Lhex-1) &&
        x<=XDP+1-(Lhex-1)+(Lhex-1)){  /* f__ 2->0->6 */
x+=-Lhex+(Lhex-1)-Lhex;
y+=Lhex-1+((Lhex-1)*2+1)+(Lhex-1);
}

else if(y==Lhex &&
        x>=XDP-(Lhex-1)+Lhex &&
        x<=XDP-(Lhex-1)+Lhex+(Lhex-1)){  /* d__ 3->0->6 */  /* 3 */
x+=Lhex-1-Lhex-Lhex;
y+=2*Lhex-1+(Lhex-1)+(Lhex-1);
}
else if(x==5*Lhex-1 &&
        y>=YDP+1-(Lhex-1) &&
        y<=YDP+1-(Lhex-1)+(Lhex-1)){  /* a__ 3->0->1 */
x+=-(2*Lhex-1)-Lhex-((Lhex-1)*2+1);
y+=-Lhex+(Lhex-1)-((Lhex-1)*1+1);
}
else if(y==x-3*Lhex+1 &&
        x>=XDP+1+Lhex &&
        x<=XDP+1+Lhex+(Lhex-1)){  /* f__ 3->0->5 */
x+=-Lhex-Lhex+(Lhex-1);
y+=Lhex-1+(Lhex-1)+((Lhex-1)*2+1);
}

else if(y==5*Lhex-1 &&
        x<=XDP+(Lhex-1)+((Lhex-1)*2+1) &&
        x>=XDP+(Lhex-1)+((Lhex-1)*2+1)-(Lhex-1)){  /* c__ 4->0->2 */  /* 4 */
x+=-(Lhex-1)-((Lhex-1)*2+1)-(Lhex-1);
y+=-(2*Lhex-1)-Lhex-((Lhex-1)*2+1);
}
else if(x==6*Lhex-2 &&
        y>=YDP+1+Lhex &&
        y<=YDP+1+Lhex+(Lhex-1)){  /* a__ 4->0->6 */
x+=-(2*Lhex-1)-((Lhex-1)*2+1)-Lhex;
```

```
y+=-Lhex-Lhex+(Lhex-1);
}
else if(y==x-2*Lhex+1 &&
        x>=XDP+1+((Lhex-1)*2+1) &&
        x<=XDP+1+((Lhex-1)*2+1)+(Lhex-1)){  /* f__ 4->0->1 */
x+=-Lhex-((Lhex-1)*2+1)-((Lhex-1)*2+1);
y+=Lhex-1-Lhex-((Lhex-1)*1+1);
}

else if(y==6*Lhex-2 &&
        x<=XDP+(Lhex-1)+(Lhex-1) &&
        x>=XDP+(Lhex-1)+(Lhex-1)-(Lhex-1)){  /* c__ 5->0->1 */  /* 5 */
x+=-(Lhex-1)-(Lhex-1)-((Lhex-1)*2+1);
y+=-(2*Lhex-1)-((Lhex-1)*2+1)-((Lhex-1)*1+1);
}
else if(x==5*Lhex-2 &&
        y>=YDP+1+((Lhex-1)*2+1) &&
        y<=YDP+1+((Lhex-1)*2+1)+(Lhex-1)){  /* a__ 5->0->2 */
x+=-(2*Lhex-1)-(Lhex-1)-(Lhex-1);
y+=-Lhex-((Lhex-1)*2+1)-((Lhex-1)*2+1);
}
else if(y==x+2*Lhex &&
        x<=XDP-1+(Lhex-1) &&
        x>=XDP-1+(Lhex-1)-(Lhex-1)){ /* e__ 5->0->3 */
x+=-(-Lhex)-(Lhex-1)+Lhex;
y+=-(Lhex-1)-((Lhex-1)*2+1)-(Lhex-1);
}

else if(y==5*Lhex-2 &&
        x<=XDP+(Lhex-1)-Lhex &&
        x>=XDP+(Lhex-1)-Lhex-(Lhex-1)){  /* c__ 6->0->3 */  /* 6 */
x+=-(Lhex-1)+Lhex+Lhex;
y+=-(2*Lhex-1)-(Lhex-1)-(Lhex-1);
}
else if(x==Lhex-1 &&
        y<=YDP-1+(Lhex-1) &&
        y>=YDP-1+(Lhex-1)-(Lhex-1)){  /* b__ 6->0->4 */
x+=2*Lhex-1+Lhex+((Lhex-1)*2+1);
y+=Lhex-(Lhex-1)+Lhex;
}
else if(y==x+3*Lhex-1 &&
        x<=XDP-1-Lhex &&
        x>=XDP-1-Lhex-(Lhex-1)){ /* e__ 6->0->2 */
x+=-(-Lhex)+Lhex-(Lhex-1);
y+=-(Lhex-1)-(Lhex-1)-((Lhex-1)*2+1);
}
```

```
else{                              /* in */
x=x;
y=y;
}
#else
if((x<0)||(x>RESO-1)||(y<0)||(y>RESO-1)){
if(c_trans==0) return /*0*/wall;
else           return /*-2*/wall;
}
#endif

if(c_trans==0) return pixel[x][y];
else{
if(!zeroFill)   return pixel_[x][y];
else            return pixel[x][y];
}
}/** getpixel_ **/


int jump(int i,int x,int y)
{
#if HEX_or_SQR==0
if(y==Lhex-1 &&
   x>=XDP-(Lhex-1)-((Lhex-1)*2+1) &&
   x<=XDP-(Lhex-1)-((Lhex-1)*2+1)+(Lhex-1)){  /* d__ 1->0->5 */  /* 1 */
nx[i]+=Lhex-1+((Lhex-1)*2+1)+(Lhex-1);
ny[i]+=2*Lhex-1+((Lhex-1)*1+1)+((Lhex-1)*2+1);

nx_[i]+=Lhex-1+((Lhex-1)*2+1)+(Lhex-1);
ny_[i]+=2*Lhex-1+((Lhex-1)*1+1)+((Lhex-1)*2+1);
}
else if(x==0 &&
        y<=YDP-1-((Lhex-1)*1+1) &&
        y>=YDP-1-((Lhex-1)*1+1)-(Lhex-1)){  /* b__ 1->0->3 */
nx[i]+=2*Lhex-1+((Lhex-1)*2+1)+Lhex;
ny[i]+=Lhex+((Lhex-1)*1+1)-(Lhex-1);

nx_[i]+=2*Lhex-1+((Lhex-1)*2+1)+Lhex;
ny_[i]+=Lhex+((Lhex-1)*1+1)-(Lhex-1);
}
else if(y==x+2*Lhex-1 &&
        x<=XDP-1-((Lhex-1)*2+1) &&
        x>=XDP-1-((Lhex-1)*2+1)-(Lhex-1)){ /* e__ 1->0->4 */
nx[i]+=-(-Lhex)+((Lhex-1)*2+1)+((Lhex-1)*2+1);
ny[i]+=-(Lhex-1)+((Lhex-1)*1+1)+Lhex;
```

```
nx_[i]+=-(-Lhex)+((Lhex-1)*2+1)+((Lhex-1)*2+1);
ny_[i]+=-(Lhex-1)+((Lhex-1)*1+1)+Lhex;
}

else if(y==0 &&
        x>=XDP-(Lhex-1)-(Lhex-1) &&
        x<=XDP-(Lhex-1)-(Lhex-1)+(Lhex-1)){  /* d__ 2->0->4 */  /* 2 */
nx[i]+=Lhex-1+(Lhex-1)+((Lhex-1)*2+1);
ny[i]+=2*Lhex-1+((Lhex-1)*2+1)+Lhex;

nx_[i]+=Lhex-1+(Lhex-1)+((Lhex-1)*2+1);
ny_[i]+=2*Lhex-1+((Lhex-1)*2+1)+Lhex;
}
else if(x==Lhex &&
        y<=YDP-1-((Lhex-1)*2+1) &&
        y>=YDP-1-((Lhex-1)*2+1)-(Lhex-1)){  /* b__ 2->0->5 */
nx[i]+=2*Lhex-1+(Lhex-1)+(Lhex-1);
ny[i]+=Lhex+((Lhex-1)*2+1)+((Lhex-1)*2+1);

nx_[i]+=2*Lhex-1+(Lhex-1)+(Lhex-1);
ny_[i]+=Lhex+((Lhex-1)*2+1)+((Lhex-1)*2+1);
}
else if(y==x-2*Lhex &&
        x>=XDP+1-(Lhex-1) &&
        x<=XDP+1-(Lhex-1)+(Lhex-1)){  /* f__ 2->0->6 */
nx[i]+=-Lhex+(Lhex-1)-Lhex;
ny[i]+=Lhex-1+((Lhex-1)*2+1)+(Lhex-1);

nx_[i]+=-Lhex+(Lhex-1)-Lhex;
ny_[i]+=Lhex-1+((Lhex-1)*2+1)+(Lhex-1);
}

else if(y==Lhex &&
        x>=XDP-(Lhex-1)+Lhex &&
        x<=XDP-(Lhex-1)+Lhex+(Lhex-1)){  /* d__ 3->0->6 */  /* 3 */
nx[i]+=Lhex-1-Lhex-Lhex;
ny[i]+=2*Lhex-1+(Lhex-1)+(Lhex-1);

nx_[i]+=Lhex-1-Lhex-Lhex;
ny_[i]+=2*Lhex-1+(Lhex-1)+(Lhex-1);
}
else if(x==5*Lhex-1 &&
        y>=YDP+1-(Lhex-1) &&
        y<=YDP+1-(Lhex-1)+(Lhex-1)){  /* a__ 3->0->1 */
nx[i]+=-(2*Lhex-1)-Lhex-((Lhex-1)*2+1);
```

```
ny[i]+=-Lhex+(Lhex-1)-((Lhex-1)*1+1);

nx_[i]+=-(2*Lhex-1)-Lhex-((Lhex-1)*2+1);
ny_[i]+=-Lhex+(Lhex-1)-((Lhex-1)*1+1);
}
else if(y==x-3*Lhex+1 &&
        x>=XDP+1+Lhex &&
        x<=XDP+1+Lhex+(Lhex-1)){  /* f__ 3->0->5 */
nx[i]+=-Lhex-Lhex+(Lhex-1);
ny[i]+=Lhex-1+(Lhex-1)+((Lhex-1)*2+1);

nx_[i]+=-Lhex-Lhex+(Lhex-1);
ny_[i]+=Lhex-1+(Lhex-1)+((Lhex-1)*2+1);
}

else if(y==5*Lhex-1 &&
        x<=XDP+(Lhex-1)+((Lhex-1)*2+1) &&
        x>=XDP+(Lhex-1)+((Lhex-1)*2+1)-(Lhex-1)){  /* c__ 4->0->2 */  /* 4 */
nx[i]+=-(Lhex-1)-((Lhex-1)*2+1)-(Lhex-1);
ny[i]+=-(2*Lhex-1)-Lhex-((Lhex-1)*2+1);

nx_[i]+=-(Lhex-1)-((Lhex-1)*2+1)-(Lhex-1);
ny_[i]+=-(2*Lhex-1)-Lhex-((Lhex-1)*2+1);
}
else if(x==6*Lhex-2 &&
        y>=YDP+1+Lhex &&
        y<=YDP+1+Lhex+(Lhex-1)){  /* a__ 4->0->6 */
nx[i]+=-(2*Lhex-1)-((Lhex-1)*2+1)-Lhex;
ny[i]+=-Lhex-Lhex+(Lhex-1);

nx_[i]+=-(2*Lhex-1)-((Lhex-1)*2+1)-Lhex;
ny_[i]+=-Lhex-Lhex+(Lhex-1);
}
else if(y==x-2*Lhex+1 &&
        x>=XDP+1+((Lhex-1)*2+1) &&
        x<=XDP+1+((Lhex-1)*2+1)+(Lhex-1)){  /* f__ 4->0->1 */
nx[i]+=-Lhex-((Lhex-1)*2+1)-((Lhex-1)*2+1);
ny[i]+=Lhex-1-Lhex-((Lhex-1)*1+1);

nx_[i]+=-Lhex-((Lhex-1)*2+1)-((Lhex-1)*2+1);
ny_[i]+=Lhex-1-Lhex-((Lhex-1)*1+1);
}

else if(y==6*Lhex-2 &&
        x<=XDP+(Lhex-1)+(Lhex-1) &&
        x>=XDP+(Lhex-1)+(Lhex-1)-(Lhex-1)){  /* c__ 5->0->1 */  /* 5 */
```

```
nx[i]+=-(Lhex-1)-(Lhex-1)-((Lhex-1)*2+1);
ny[i]+=-(2*Lhex-1)-((Lhex-1)*2+1)-((Lhex-1)*1+1);

nx_[i]+=-(Lhex-1)-(Lhex-1)-((Lhex-1)*2+1);
ny_[i]+=-(2*Lhex-1)-((Lhex-1)*2+1)-((Lhex-1)*1+1);
}
else if(x==5*Lhex-2 &&
        y>=YDP+1+((Lhex-1)*2+1) &&
        y<=YDP+1+((Lhex-1)*2+1)+(Lhex-1)){  /* a__ 5->0->2 */
nx[i]+=-(2*Lhex-1)-(Lhex-1)-(Lhex-1);
ny[i]+=-Lhex-((Lhex-1)*2+1)-((Lhex-1)*2+1);

nx_[i]+=-(2*Lhex-1)-(Lhex-1)-(Lhex-1);
ny_[i]+=-Lhex-((Lhex-1)*2+1)-((Lhex-1)*2+1);
}
else if(y==x+2*Lhex &&
        x<=XDP-1+(Lhex-1) &&
        x>=XDP-1+(Lhex-1)-(Lhex-1)){ /* e__ 5->0->3 */
nx[i]+=-(-Lhex)-(Lhex-1)+Lhex;
ny[i]+=-(Lhex-1)-((Lhex-1)*2+1)-(Lhex-1);

nx_[i]+=-(-Lhex)-(Lhex-1)+Lhex;
ny_[i]+=-(Lhex-1)-((Lhex-1)*2+1)-(Lhex-1);
}

else if(y==5*Lhex-2 &&
        x<=XDP+(Lhex-1)-Lhex &&
        x>=XDP+(Lhex-1)-Lhex-(Lhex-1)){  /* c__ 6->0->3 */  /* 6 */
nx[i]+=-(Lhex-1)+Lhex+Lhex;
ny[i]+=-(2*Lhex-1)-(Lhex-1)-(Lhex-1);

nx_[i]+=-(Lhex-1)+Lhex+Lhex;
ny_[i]+=-(2*Lhex-1)-(Lhex-1)-(Lhex-1);
}
else if(x==Lhex-1 &&
        y<=YDP-1+(Lhex-1) &&
        y>=YDP-1+(Lhex-1)-(Lhex-1)){  /* b__ 6->0->4 */
nx[i]+=2*Lhex-1+Lhex+((Lhex-1)*2+1);
ny[i]+=Lhex-(Lhex-1)+Lhex;

nx_[i]+=2*Lhex-1+Lhex+((Lhex-1)*2+1);
ny_[i]+=Lhex-(Lhex-1)+Lhex;
}
else if(y==x+3*Lhex-1 &&
        x<=XDP-1-Lhex &&
        x>=XDP-1-Lhex-(Lhex-1)){ /* e__ 6->0->2 */
```

```
nx[i]+=-(-Lhex)+Lhex-(Lhex-1);
ny[i]+=-(Lhex-1)-(Lhex-1)-((Lhex-1)*2+1);

nx_[i]+=-(-Lhex)+Lhex-(Lhex-1);
ny_[i]+=-(Lhex-1)-(Lhex-1)-((Lhex-1)*2+1);
}

else{                               /* in */
nx[i]=nx[i];
ny[i]=ny[i];

nx_[i]=nx_[i];
ny_[i]=ny_[i];
}
#else
if(nx[i]==-1) {nx[i]=RESO-1;nx_[i]=RESO;}  /* modification of b,S */
else if(nx[i]==RESO) {nx[i]=0;nx_[i]=-1;}
if(ny[i]==-1) {ny[i]=RESO-1;ny_[i]=RESO;}
else if(ny[i]==RESO) {ny[i]=0;ny_[i]=-1;}
#endif

return 0;
}/** jump **/


int connect_nxpm(int nxpm)
{
#if HEX_or_SQR==0
return nxpm;
#else
if(nxpm==-1) return RESO-1;
else if(nxpm==RESO) return 0;
else return nxpm;
#endif

}/** connect_nxpm **/


int connect_nypm(int nypm)
{
#if HEX_or_SQR==0
return nypm;
#else
if(nypm==-1) return RESO-1;
else if(nypm==RESO) return 0;
else return nypm;
```

```
#endif
}/** connect_nypm **/


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;/*printf(" ?\n");*/}
}/** 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 **/


int random_(int n)
{
int val;

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

return val;
}/** random_ **/


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 **/



#if HEX_or_SQR==0
int cag_r(int nax_,int nay_,int color_)
{
int i,j;
int flag_[CPMAX],flag_pp_[CPMAX];
int nax[CPMAX],nay[CPMAX];
int ca,acolor[CPMAX];
int cp,algo,ssize,bdrflag;
long fpos[CPMAX];
double val,angle;

ssize=sizeof(s);
if(c_trans==0 || SPmode==1) cp=CPMAX_;else cp=1;

for(i=0;i<CPMAX_;i++){
rcount[i]=0;
fp_mem[i]=0;
flag_[i]=1;
}

ca=16;

/*999*/
if(c_trans==0 || SPmode==1){
if(CPGRP==2){
nax[0]=XDP-(Lhex-1)+d3                ;nay[0]=YDP;  /* 0 */
nax[1]=XDP+(Lhex-1)-d3                ;nay[1]=YDP;  /* blue */

nax[2]=XDP-(Lhex-1)+d3-((Lhex-1)*2+1) ;nay[2]=YDP           -((Lhex-1)*1+1);  /* 1 */
nax[3]=XDP+(Lhex-1)-d3-((Lhex-1)*2+1) ;nay[3]=YDP           -((Lhex-1)*1+1);
```

```
nax[4]=XDP-(Lhex-1)+d3-(Lhex-1)       ;nay[4]=YDP           -((Lhex-1)*2+1);  /* 2 */
nax[5]=XDP+(Lhex-1)-d3-(Lhex-1)       ;nay[5]=YDP           -((Lhex-1)*2+1);

nax[6]=XDP-(Lhex-1)+d3+Lhex           ;nay[6]=YDP           -(Lhex-1);  /* 3 */
nax[7]=XDP+(Lhex-1)-d3+Lhex           ;nay[7]=YDP           -(Lhex-1);

nax[8]=XDP-(Lhex-1)+d3+((Lhex-1)*2+1) ;nay[8]=YDP           +Lhex;  /* 4 */
nax[9]=XDP+(Lhex-1)-d3+((Lhex-1)*2+1) ;nay[9]=YDP           +Lhex;

nax[10]=XDP-(Lhex-1)+d3+(Lhex-1)      ;nay[10]=YDP          +((Lhex-1)*2+1);  /* 5 */
nax[11]=XDP+(Lhex-1)-d3+(Lhex-1)      ;nay[11]=YDP           +((Lhex-1)*2+1);

nax[12]=XDP-(Lhex-1)+d3-Lhex          ;nay[12]=YDP           +(Lhex-1);  /* 6 */
nax[13]=XDP+(Lhex-1)-d3-Lhex          ;nay[13]=YDP           +(Lhex-1);
}
else{
nax[0]=XDP-(Lhex-1)+d3                ;nay[0]=YDP;  /* 0 */
nax[3]=XDP+(Lhex-1)-d3                ;nay[3]=YDP;  /* blue */
nax[2]=XDP                            ;nay[2]=YDP-(Lhex-1)+d3;  /* green */
nax[5]=XDP                            ;nay[5]=YDP+(Lhex-1)-d3;  /* sky */
nax[1]=XDP-(Lhex-1)+d3                ;nay[1]=YDP-(Lhex-1)+d3;  /* red */
nax[4]=XDP+(Lhex-1)-d3                ;nay[4]=YDP+(Lhex-1)-d3;  /* magenta */

nax[6]=XDP-(Lhex-1)+d3-((Lhex-1)*2+1) ;nay[6]=YDP           -((Lhex-1)*1+1);  /* 1 */
nax[9]=XDP+(Lhex-1)-d3-((Lhex-1)*2+1) ;nay[9]=YDP           -((Lhex-1)*1+1);
nax[8]=XDP           -((Lhex-1)*2+1)  ;nay[8]=YDP-(Lhex-1)+d3-((Lhex-1)*1+1);
nax[11]=XDP           -((Lhex-1)*2+1) ;nay[11]=YDP+(Lhex-1)-d3-((Lhex-1)*1+1);
nax[7]=XDP-(Lhex-1)+d3-((Lhex-1)*2+1) ;nay[7]=YDP-(Lhex-1)+d3-((Lhex-1)*1+1);
nax[10]=XDP+(Lhex-1)-d3-((Lhex-1)*2+1);nay[10]=YDP+(Lhex-1)-d3-((Lhex-1)*1+1);

nax[12]=XDP-(Lhex-1)+d3-(Lhex-1)      ;nay[12]=YDP          -((Lhex-1)*2+1);  /* 2 */
nax[15]=XDP+(Lhex-1)-d3-(Lhex-1)      ;nay[15]=YDP           -((Lhex-1)*2+1);
nax[14]=XDP           -(Lhex-1)       ;nay[14]=YDP-(Lhex-1)+d3-((Lhex-1)*2+1);
nax[17]=XDP           -(Lhex-1)       ;nay[17]=YDP+(Lhex-1)-d3-((Lhex-1)*2+1);
nax[13]=XDP-(Lhex-1)+d3-(Lhex-1)      ;nay[13]=YDP-(Lhex-1)+d3-((Lhex-1)*2+1);
nax[16]=XDP+(Lhex-1)-d3-(Lhex-1)      ;nay[16]=YDP+(Lhex-1)-d3-((Lhex-1)*2+1);

nax[18]=XDP-(Lhex-1)+d3+Lhex          ;nay[18]=YDP           -(Lhex-1);  /* 3 */
nax[21]=XDP+(Lhex-1)-d3+Lhex          ;nay[21]=YDP           -(Lhex-1);
nax[20]=XDP           +Lhex           ;nay[20]=YDP-(Lhex-1)+d3-(Lhex-1);
nax[23]=XDP           +Lhex           ;nay[23]=YDP+(Lhex-1)-d3-(Lhex-1);
nax[19]=XDP-(Lhex-1)+d3+Lhex          ;nay[19]=YDP-(Lhex-1)+d3-(Lhex-1);
nax[22]=XDP+(Lhex-1)-d3+Lhex          ;nay[22]=YDP+(Lhex-1)-d3-(Lhex-1);

nax[24]=XDP-(Lhex-1)+d3+((Lhex-1)*2+1);nay[24]=YDP           +Lhex;  /* 4 */
```

```
nax[27]=XDP+(Lhex-1)-d3+((Lhex-1)*2+1);nay[27]=YDP           +Lhex;
nax[26]=XDP           +((Lhex-1)*2+1) ;nay[26]=YDP-(Lhex-1)+d3+Lhex;
nax[29]=XDP           +((Lhex-1)*2+1) ;nay[29]=YDP+(Lhex-1)-d3+Lhex;
nax[25]=XDP-(Lhex-1)+d3+((Lhex-1)*2+1);nay[25]=YDP-(Lhex-1)+d3+Lhex;
nax[28]=XDP+(Lhex-1)-d3+((Lhex-1)*2+1);nay[28]=YDP+(Lhex-1)-d3+Lhex;

nax[30]=XDP-(Lhex-1)+d3+(Lhex-1)      ;nay[30]=YDP          +((Lhex-1)*2+1);  /* 5 */
nax[33]=XDP+(Lhex-1)-d3+(Lhex-1)      ;nay[33]=YDP           +((Lhex-1)*2+1);
nax[32]=XDP           +(Lhex-1)       ;nay[32]=YDP-(Lhex-1)+d3+((Lhex-1)*2+1);
nax[35]=XDP           +(Lhex-1)       ;nay[35]=YDP+(Lhex-1)-d3+((Lhex-1)*2+1);
nax[31]=XDP-(Lhex-1)+d3+(Lhex-1)      ;nay[31]=YDP-(Lhex-1)+d3+((Lhex-1)*2+1);
nax[34]=XDP+(Lhex-1)-d3+(Lhex-1)      ;nay[34]=YDP+(Lhex-1)-d3+((Lhex-1)*2+1);

nax[36]=XDP-(Lhex-1)+d3-Lhex          ;nay[36]=YDP           +(Lhex-1);  /* 6 */
nax[39]=XDP+(Lhex-1)-d3-Lhex          ;nay[39]=YDP           +(Lhex-1);
nax[38]=XDP           -Lhex           ;nay[38]=YDP-(Lhex-1)+d3+(Lhex-1);
nax[41]=XDP           -Lhex           ;nay[41]=YDP+(Lhex-1)-d3+(Lhex-1);
nax[37]=XDP-(Lhex-1)+d3-Lhex          ;nay[37]=YDP-(Lhex-1)+d3+(Lhex-1);
nax[40]=XDP+(Lhex-1)-d3-Lhex          ;nay[40]=YDP+(Lhex-1)-d3+(Lhex-1);
}

if(1){
/* 6 colors */
acolor[0]=9;acolor[1]=10;  /* 0 */
acolor[2]=11;acolor[3]=12;
acolor[4]=13;acolor[5]=14;

acolor[6]=9;acolor[7]=/*10*/7;  /* 1 */
acolor[8]=11;acolor[9]=12;
acolor[10]=13;acolor[11]=14;

acolor[12]=9;acolor[13]=10;  /* 2 */
acolor[14]=11;acolor[15]=12;
acolor[16]=13;acolor[17]=14;

acolor[18]=9;acolor[19]=10;  /* 3 */
acolor[20]=11;acolor[21]=12;
acolor[22]=13;acolor[23]=14;

acolor[24]=9;acolor[25]=10;  /* 4 */
acolor[26]=11;acolor[27]=12;
acolor[28]=13;acolor[29]=14;

acolor[30]=9;acolor[31]=10;  /* 5 */
acolor[32]=11;acolor[33]=12;
acolor[34]=13;acolor[35]=14;
```

```
acolor[36]=9;acolor[37]=10;  /* 6 */
acolor[38]=11;acolor[39]=12;
acolor[40]=13;acolor[41]=14;
}
else{
/* 42 colors */
acolor[0]=0;acolor[1]=1;  /* 0 */
acolor[2]=2;acolor[3]=3;
acolor[4]=4;acolor[5]=5;

acolor[6]=6;acolor[7]=7;  /* 1 */
acolor[8]=8;acolor[9]=9;
acolor[10]=10;acolor[11]=11;

acolor[12]=12;acolor[13]=13;  /* 2 */
acolor[14]=14;acolor[15]=15;
acolor[16]=17;acolor[17]=18;

acolor[18]=19;acolor[19]=20;  /* 3 */
acolor[20]=21;acolor[21]=22;
acolor[22]=23;acolor[23]=24;

acolor[24]=25;acolor[25]=26;  /* 4 */
acolor[26]=27;acolor[27]=28;
acolor[28]=29;acolor[29]=30;

acolor[30]=31;acolor[31]=32;  /* 5 */
acolor[32]=33;acolor[33]=34;
acolor[34]=35;acolor[35]=36;

acolor[36]=37;acolor[37]=38;  /* 6 */
acolor[38]=39;acolor[39]=40;
acolor[40]=41;acolor[41]=42;
}
}
else{                                  /* C_and_S_old */
nax[0]=nax_;nay[0]=nay_;
acolor[0]=color_;
}

if(acolor[0]!=ca){
if(/*(c1!=ca)||(c2!=ca)||(c3!=ca)||(c4!=ca)||(c5!=ca)||(c7!=ca)*/0){
beep(2000);
cp=0;
}/**if(c1,c2,c3,c4)**/
```

```
else{
bdrflag=0;

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]);
rcount[i]++;

if(c_trans==1 && SPmode==1 && !bdrflag){
for(j=0;j<bdrnum;j++)
if(nx[i]==bdr[j].cx && ny[i]==bdr[j].cy) {bdrflag=1;break;}
}
else if(c_trans==2 && !SPmode && !tmp4){
searchflag=1;
search_i(nx[0],ny[0],acolor[0]);
searchflag=0;
}
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX_) break;}
else break;
}/**while(1)**/

if(c_trans==1 && SPmode==1 && bdrnum && bdrflag){
for(i=0;i<CPMAX_;i++) {bdrs[bdrsnum+i].cx=nx[i];bdrs[bdrsnum+i].cy=ny[i];}
bdrsnum+=CPMAX_;
}

if(/*(c1!=ca)||(c2!=ca)||(c3!=ca)||(c4!=ca)||(c5!=ca)||(c7!=ca)*/0){
beep(2000);
cp=0;
}
else{
bdrflag=0;

i=0;
while(1){
if(flag_[i]){                        /* CP_? */
nx_[i]=nax[i];ny_[i]=nay[i];
ig=i;
if(c_trans==0 || SPmode==1){
if(CPGRP==2){
     if(i%2==0) nax[i]++;
else if(i%2==1) nax[i]--;
}
else{
     if(i%6==0) {nax[i]++;nay[i]++;}
else if(i%6==1) nay[i]++;
else if(i%6==2) nax[i]--;
else if(i%6==3) {nax[i]--;nay[i]--;}
else if(i%6==4) nay[i]--;
else if(i%6==5) nax[i]++;
}
}
else{                                    /* C_and_S_old */
nax[0]++;
}

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

jump(i,nx[i],ny[i]);                     /* modification of b,S */

putpixel(nx[i],ny[i],acolor[i]);
rcount[i]++;

if(c_trans==1 && SPmode==1 && !bdrflag){
for(j=0;j<bdrnum;j++)
if(nx[i]==bdr[j].cx && ny[i]==bdr[j].cy) {bdrflag=1;break;}
}
else if(c_trans==2 && !SPmode && !tmp4){
searchflag=1;
search_i(nx[0],ny[0],acolor[0]);
searchflag=0;
}
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX_) break;}
else break;
}/**while(1)**/

if(c_trans==1 && SPmode==1 && bdrnum && bdrflag){
for(i=0;i<CPMAX_;i++) {bdrs[bdrsnum+i].cx=nx[i];bdrs[bdrsnum+i].cy=ny[i];}
bdrsnum+=CPMAX_;
}
}/**else(c1,c2,c3,c4,c5,c7)**/
```

```
}/**else(c1,c2,c3,c4,c5,c7)**/
}/**if(acolor)**/
else{
cp=0;
}/**else(acolor)**/

if(c_trans>0 && SPmode!=1){
}

if(YOUR_ART==0 && c_trans==0 && GRPH==1 && cnt==0) use_subroop();
cnt++;
/***************************** while(cp) -> *****************************/

while(cp){
kbhit_();
if(refill==0) break;
if(0) use_subroop();

if(random_(2)==0) algo=0;else algo=1;
bdrflag=0;

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;
nxp_c=connect_nxpm(nxp);nxm_c=connect_nxpm(nxm);
nyp_c=connect_nypm(nyp);nym_c=connect_nypm(nym);
c1=getpixel_(nxp_c,ny[i]);c2=getpixel_(nx[i],nyp_c);c3=getpixel_(nxm_c,ny[i]);
c4=getpixel_(nx[i],nym_c);
c5=getpixel_(nxp_c,nyp_c);c7=getpixel_(nxm_c,nym_c);

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 */

                                       /* new S */
if(algo==0 || YOUR_ART==1){
val=-5;
if((c1!=ca)&&(c5==ca)){  /* CW (no phase) */
  if((angle=getangle(nxp,nyp))>val) {val=angle;nx[i]=nxp;ny[i]=nyp;}}
if((c5!=ca)&&(c2==ca)){
  if((angle=getangle(std_x,nyp))>val) {val=angle;nx[i]=std_x;ny[i]=nyp;}}
if((c2!=ca)&&(c3==ca)){
  if((angle=getangle(nxm,std_y))>val) {val=angle;nx[i]=nxm;ny[i]=std_y;}}
if((c3!=ca)&&(c7==ca)){
  if((angle=getangle(nxm,nym))>val) {val=angle;nx[i]=nxm;ny[i]=nym;}}
if((c7!=ca)&&(c4==ca)){
  if((angle=getangle(std_x,nym))>val) {val=angle;nx[i]=std_x;ny[i]=nym;}}
if((c4!=ca)&&(c1==ca)){
  if((angle=getangle(nxp,std_y))>val) {val=angle;nx[i]=nxp;ny[i]=std_y;}}
if(val==-5) /*beep(10000)*/;
}/**if(switching,algorithm,al[i])**/
else{
val=5;
if((c1!=ca)&&(c4==ca)){  /* CCW (no phase) */
  if((angle=getangle(std_x,nym))<val) {val=angle;nx[i]=std_x;ny[i]=nym;}}
if((c4!=ca)&&(c7==ca)){
  if((angle=getangle(nxm,nym))<val) {val=angle;nx[i]=nxm;ny[i]=nym;}}
if((c7!=ca)&&(c3==ca)){
  if((angle=getangle(nxm,std_y))<val) {val=angle;nx[i]=nxm;ny[i]=std_y;}}
if((c3!=ca)&&(c2==ca)){
  if((angle=getangle(std_x,nyp))<val) {val=angle;nx[i]=std_x;ny[i]=nyp;}}
if((c2!=ca)&&(c5==ca)){
  if((angle=getangle(nxp,nyp))<val) {val=angle;nx[i]=nxp;ny[i]=nyp;}}
if((c5!=ca)&&(c1==ca)){
  if((angle=getangle(nxp,std_y))<val) {val=angle;nx[i]=nxp;ny[i]=std_y;}}
if(val==5) /*beep(10000)*/;
}/**else(switching,algorithm,al[i])**/

jump(i,nx[i],ny[i]);                /* modification of b,S */

putpixel(nx[i],ny[i],acolor[i]);
rcount[i]++;

if(c_trans==1 && SPmode==1 && !bdrflag){
for(j=0;j<bdrnum;j++)
if(nx[i]==bdr[j].cx && ny[i]==bdr[j].cy) {bdrflag=1;break;}
}
else if(c_trans==2 && !SPmode && !tmp4){
searchflag=1;
search_i(nx[0],ny[0],acolor[0]);
searchflag=0;
}

flag_pp_[i]=1;
}/**if(c1,c2,c3,c4)**/
else{
```

```
flag_pp_[i]=0;

if(ftell_mem(i)==0) {flag_[i]=0;cp--;if(cp==0) break;}  /* 2pie/3 */ /* pie */
fread_mem(i);
nx[i]=s.xx;ny[i]=s.yy;nx_[i]=s.xx_;ny_[i]=s.yy_;
}/**else(c1,c2,c3,c4)**/
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX_) break;}
else break;
}/**while(1)**/

if(flag_pp_[0]==1){
if(c_trans==1 && SPmode==1 && bdrnum && bdrflag){
for(i=0;i<CPMAX_;i++) {bdrs[bdrsnum+i].cx=nx[i];bdrs[bdrsnum+i].cy=ny[i];}
bdrsnum+=CPMAX_;
}
}

if(flag_pp_[0]==0){
}
}/**while(cp)**/

/*printf(" cag_r:%ld\n",rcount[0]);*/
return 0;
}/** cag_r **/
#else
int cag_r(int nax_,int nay_,int color_)
{
int i,j;
int flag_[CPMAX],flag_pp_[CPMAX];
int nax[CPMAX],nay[CPMAX];
int ca,acolor[CPMAX];
int cp,algo,ssize,bdrflag;
long fpos[CPMAX];
double val,angle;

ssize=sizeof(s);
if(c_trans==0 || SPmode==1) cp=CPMAX_;else cp=1;

for(i=0;i<CPMAX_;i++){
rcount[i]=0;
fp_mem[i]=0;
flag_[i]=1;
}
```

```
ca=16;

/*999*/
if(c_trans==0 || SPmode==1){
nax[0]=Lsqr*1+0                ;nay[0]=Lsqr*1+0;
nax[1]=Lsqr*3+1                ;nay[1]=Lsqr*1+0;
nax[2]=Lsqr*5+2                ;nay[2]=Lsqr*1+0;
nax[3]=Lsqr*7+3               ;nay[3]=Lsqr*1+0;

nax[4]=Lsqr*1+0               ;nay[4]=Lsqr*3+1;
nax[5]=Lsqr*3+1               ;nay[5]=Lsqr*3+1;
nax[6]=Lsqr*5+2                ;nay[6]=Lsqr*3+1;
nax[7]=Lsqr*7+3                ;nay[7]=Lsqr*3+1;

nax[8]=Lsqr*1+0                ;nay[8]=Lsqr*5+2;
nax[9]=Lsqr*3+1                ;nay[9]=Lsqr*5+2;
nax[10]=Lsqr*5+2                ;nay[10]=Lsqr*5+2;
nax[11]=Lsqr*7+3               ;nay[11]=Lsqr*5+2;

nax[12]=Lsqr*1+0               ;nay[12]=Lsqr*7+3;
nax[13]=Lsqr*3+1               ;nay[13]=Lsqr*7+3;
nax[14]=Lsqr*5+2                ;nay[14]=Lsqr*7+3;
nax[15]=Lsqr*7+3                ;nay[15]=Lsqr*7+3;

acolor[0]=1;acolor[1]=2;
acolor[2]=3;acolor[3]=4;

acolor[4]=5;acolor[5]=6;
acolor[6]=7;acolor[7]=8;



acolor[8]=9;acolor[9]=10;
acolor[10]=11;acolor[11]=12;

acolor[12]=13;acolor[13]=14;
acolor[14]=15;acolor[15]=17;
}
else{                              /* c_trans>0 && SPmode!=1 */
nax[0]=nax_;nay[0]=nay_;
acolor[0]=color_;
}

if(acolor[0]!=ca){
if(/*(c1!=ca)||(c2!=ca)||(c3!=ca)||(c4!=ca)||(c5!=ca)||(c7!=ca)*/0){
beep(2000);
```

```
cp=0;
}/**if(c1,c2,c3,c4)**/
else{
bdrflag=0;

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]);
rcount[i]++;

if(c_trans==1 && SPmode==1 && !bdrflag){
for(j=0;j<bdrnum;j++)
if(nx[i]==bdr[j].cx && ny[i]==bdr[j].cy) {bdrflag=1;break;}
}
else if(c_trans==2 && !SPmode && !tmp4){
searchflag=1;
search_i(nx[0],ny[0],acolor[0]);
searchflag=0;
}
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX_) break;}
else break;
}/**while(1)**/

if(c_trans==1 && SPmode==1 && bdrnum && bdrflag){
for(i=0;i<CPMAX_;i++) {bdrs[bdrsnum+i].cx=nx[i];bdrs[bdrsnum+i].cy=ny[i];}
bdrsnum+=CPMAX_;
}

if(/*(c1!=ca)||(c2!=ca)||(c3!=ca)||(c4!=ca)||(c5!=ca)||(c7!=ca)*/0){
beep(2000);
cp=0;
}
else{
bdrflag=0;

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

if(c_trans==0 || SPmode==1){
if(PHASE==0) nax[i]++;
else if(PHASE==1){
if((i/4)%2==0) nax[i]++;
else           nax[i]--;
}
else{
if((i/4)%2==0){
if(i%2==0) nax[i]++;
else       nax[i]--;
}
else{
if(i%2==0) nax[i]--;
else       nax[i]++;
}
}
}
else{                              /* C_and_S_old */
nax[0]++;
}

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

jump(i,nx[i],ny[i]);               /* modification of b,S */

putpixel(nx[i],ny[i],acolor[i]);
rcount[i]++;

if(c_trans==1 && SPmode==1 && !bdrflag){
for(j=0;j<bdrnum;j++)
if(nx[i]==bdr[j].cx && ny[i]==bdr[j].cy) {bdrflag=1;break;}
}
else if(c_trans==2 && !SPmode && !tmp4){
searchflag=1;
search_i(nx[0],ny[0],acolor[0]);
searchflag=0;
}
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX_) break;}
else break;
}/**while(1)**/
```

```
if(c_trans==1 && SPmode==1 && bdrnum && bdrflag){
for(i=0;i<CPMAX_;i++) {bdrs[bdrsnum+i].cx=nx[i];bdrs[bdrsnum+i].cy=ny[i];}
bdrsnum+=CPMAX_;
}
}/**else(c1,c2,c3,c4)**/
}/**else(c1,c2,c3,c4)**/
}/**if(acolor)**/
else{
cp=0;
}/**else(acolor)**/

if(c_trans>0 && SPmode!=1){
}

if(YOUR_ART==0 && c_trans==0 && GRPH==1 && cnt==0) use_subroop();
cnt++;
/***************************** while(cp) -> *****************************/

while(cp){
kbhit_();
if(refill==0) break;
if(0) use_subroop();

if(random_(2)==0) algo=0;else algo=1;
bdrflag=0;

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;
nxp_c=connect_nxpm(nxp);nxm_c=connect_nxpm(nxm);
nyp_c=connect_nypm(nyp);nym_c=connect_nypm(nym);
c1=getpixel_(nxp_c,ny[i]);c2=getpixel_(nx[i],nyp_c);c3=getpixel_(nxm_c,ny[i]);
c4=getpixel_(nx[i],nym_c);

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 */

                                        /* new S */
if(algo==0 || YOUR_ART==1){
```

```
val=-5;
if((c1!=ca)&&/*(c5!=ca)&&*/(c2==ca)){  /* CW (no phase) */
  if((angle=getangle(std_x,nyp))>val) {val=angle;nx[i]=std_x;ny[i]=nyp;}}
if((c2!=ca)&&/*(c6!=ca)&&*/(c3==ca)){
  if((angle=getangle(nxm,std_y))>val) {val=angle;nx[i]=nxm;ny[i]=std_y;}}
if((c3!=ca)&&/*(c7!=ca)&&*/(c4==ca)){
  if((angle=getangle(std_x,nym))>val) {val=angle;nx[i]=std_x;ny[i]=nym;}}
if((c4!=ca)&&/*(c8!=ca)&&*/(c1==ca)){
  if((angle=getangle(nxp,std_y))>val) {val=angle;nx[i]=nxp;ny[i]=std_y;}}
}/**if(switching,algorithm,al[i])**/
else{
val=5;
if((c1!=ca)&&/*(c8!=ca)&&*/(c4==ca)){  /* CCW (no phase) */
  if((angle=getangle(std_x,nym))<val) {val=angle;nx[i]=std_x;ny[i]=nym;}}
if((c4!=ca)&&/*(c7!=ca)&&*/(c3==ca)){
  if((angle=getangle(nxm,std_y))<val) {val=angle;nx[i]=nxm;ny[i]=std_y;}}
if((c3!=ca)&&/*(c6!=ca)&&*/(c2==ca)){
  if((angle=getangle(std_x,nyp))<val) {val=angle;nx[i]=std_x;ny[i]=nyp;}}
if((c2!=ca)&&/*(c5!=ca)&&*/(c1==ca)){
  if((angle=getangle(nxp,std_y))<val) {val=angle;nx[i]=nxp;ny[i]=std_y;}}
}/**else(switching,algorithm,al[i])**/

jump(i,nx[i],ny[i]);                /* modification of b,S */

putpixel(nx[i],ny[i],acolor[i]);
rcount[i]++;

if(c_trans==1 && SPmode==1 && !bdrflag){
for(j=0;j<bdrnum;j++)
if(nx[i]==bdr[j].cx && ny[i]==bdr[j].cy) {bdrflag=1;break;}
}
else if(c_trans==2 && !SPmode && !tmp4){
searchflag=1;
search_i(nx[0],ny[0],acolor[0]);
searchflag=0;
}

flag_pp_[i]=1;
}/**if(c1,c2,c3,c4)**/
else{
flag_pp_[i]=0;

if(ftell_mem(i)==0) {flag_[i]=0;cp--;if(cp==0) break;}  /* 2pie/3 */ /* pie */
fread_mem(i);
nx[i]=s.xx;ny[i]=s.yy;nx_[i]=s.xx_;ny_[i]=s.yy_;
}/**else(c1,c2,c3,c4)**/
```

```
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX_) break;}
else break;
}/**while(1)**/

if(flag_pp_[0]==1){
if(c_trans==1 && SPmode==1 && bdrnum && bdrflag){
for(i=0;i<CPMAX_;i++) {bdrs[bdrsnum+i].cx=nx[i];bdrs[bdrsnum+i].cy=ny[i];}
bdrsnum+=CPMAX_;
}
}

if(flag_pp_[0]==0){
}
}/**while(cp)**/

/*printf(" cag_r:%ld\n",rcount[0]);*/
return 0;
}/** cag_r **/
#endif


void write_neighbor(char flag,int color)
{
int i=ig;
int Ccolor,old,posflag;
int nxp,nxm,nyp,nym,Nx,Ny,std_x,std_y;

     if(c_trans==1) Ccolor=/*acolor[i]*/color;
else if(c_trans==2) Ccolor=color_old_g;
old=c_trans;c_trans=2;



if(flag%2){                         /* 1, 3 */
nxp=nx[i]+1;nyp=ny[i]+1;nxm=nx[i]-1;nym=ny[i]-1;
nxp_c=connect_nxpm(nxp);nxm_c=connect_nxpm(nxm);
nyp_c=connect_nypm(nyp);nym_c=connect_nypm(nym);
c1=getpixel_(nxp_c,ny[i]);c2=getpixel_(nx[i],nyp_c);c3=getpixel_(nxm_c,ny[i]);
c4=getpixel_(nx[i],nym_c);
c5=getpixel_(nxp_c,nyp_c);c7=getpixel_(nxm_c,nym_c);
c6=getpixel_(nxm_c,nyp_c);c8=getpixel_(nxp_c,nym_c);
}
```

```
if(c1==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nxp_c;ny[i]=ny[i];

jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
if(c2==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nx[i];ny[i]=nyp_c;

jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
if(c3==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nxm_c;ny[i]=ny[i];

jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
if(c4==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nx[i];ny[i]=nym_c;

jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}

if(flag<=1){
if(c5==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nxp_c;ny[i]=nyp_c;

jump(i,nx[i],ny[i]);
```

```
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
if(c7==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nxm_c;ny[i]=nym_c;

jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
}
else{
if(c5==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nxp_c;ny[i]=nyp_c;

jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
if(c6==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nxm_c;ny[i]=nyp_c;

jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
if(c7==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nxm_c;ny[i]=nym_c;

jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
if(c8==-1){
std_x=nx[i];std_y=ny[i];

nx[i]=nxp_c;ny[i]=nym_c;
```

```
jump(i,nx[i],ny[i]);
putpixel_(nx[i],ny[i],Ccolor);
nx[i]=std_x;ny[i]=std_y;
}
}

c_trans=old;
}/** write_neighbor **/


#if HEX_or_SQR==0
int cag_r_6(int nax_,int nay_,int color_)  /* xi,yi,pcolor_old */
{
int i;
int flag_[CPMAX],flag_pp_[CPMAX];
int nax[CPMAX],nay[CPMAX];
int ca,acolor[CPMAX];
int cp,algo,ssize;
long fpos[CPMAX];
double val,angle;

ssize=sizeof(s);
cp=1;

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

if(!zeroFill){
     if(tmp4==2) ca=16;             /* C+0 */
else if(tmp4==3) ca=-1;             /* S_ */
else if(tmp4==4) ca=16;             /* 0+S */
else if(tmp4==5) ca=-1;             /* C_ */
else             ca=/*15*/16;
}
else             ca=zg;             /* S_ */

nax[0]=nax_;nay[0]=nay_;
acolor[0]=color_;

if(!zeroFill){
if(pixel_[nax[0]][nay[0]]!=ca) return 1;
}
```

```
if(acolor[0]!=ca){
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]);
rcount[i]++;

     if(tmp4==1){                /* i=0 */
bdr[bdrnum].cx=nx[i];bdr[bdrnum].cy=ny[i];
bdrnum++;
}
else if(tmp4==2) write_neighbor(1,acolor[i]);
}/**if(flag_[i])**/

i++;break;
}/**while(1)**/
}/**if(acolor[])**/
else cp=0;

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

while(cp){
if(refill==0) {refill=2;break;}

i=0;
while(1){
/*use_subroop();*/                   /* kbhit_() */
/*if(refill==0) break;*/

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

nxp=nx[i]+1;nyp=ny[i]+1;nxm=nx[i]-1;nym=ny[i]-1;
nxp_c=connect_nxpm(nxp);nxm_c=connect_nxpm(nxm);
nyp_c=connect_nypm(nyp);nym_c=connect_nypm(nym);
c1=getpixel_(nxp_c,ny[i]);c2=getpixel_(nx[i],nyp_c);c3=getpixel_(nxm_c,ny[i]);
c4=getpixel_(nx[i],nym_c);
c5=getpixel_(nxp_c,nyp_c);c7=getpixel_(nxm_c,nym_c);

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

std_x=nx[i];std_y=ny[i];         /* S */
if(tmp4==2) write_neighbor(0,acolor[i]);

  /* CW (no phase) */
     if(c1==ca) {nx[i]=nxp;ny[i]=std_y;}
else if(c5==ca) {nx[i]=nxp;ny[i]=nyp;}
else if(c2==ca) {nx[i]=std_x;ny[i]=nyp;}
else if(c3==ca) {nx[i]=nxm;ny[i]=std_y;}
else if(c7==ca) {nx[i]=nxm;ny[i]=nym;}
else if(c4==ca) {nx[i]=std_x;ny[i]=nym;}

jump(i,nx[i],ny[i]);                /* modification of b,S */

putpixel(nx[i],ny[i],acolor[i]);
rcount[i]++;

     if(tmp4==1){                   /* i=0 */
bdr[bdrnum].cx=nx[i];bdr[bdrnum].cy=ny[i];
bdrnum++;
}
else if(tmp4==2) write_neighbor(1,acolor[i]);

flag_pp_[i]=1;
}/**if(c1,c2,c3,c4)**/
else{
flag_pp_[i]=0;

if(ftell_mem(i)==0) {flag_[i]=0;cp--;if(cp==0) break;}  /* 2pie/3 */ /* pie */
fread_mem(i);
nx[i]=s.xx;ny[i]=s.yy;
}/**else(c1,c2,c3,c4)**/
}/**if(flag_[i])**/

i++;break;
}/**while(1)**/

if(flag_pp_[0]==1){
}

if(flag_pp_[0]==0){
}
}/**while(cp)**/

if((c_trans==1 && !SPmode)||(zeroFill)){
/*printf(" %ld\n",rcount[0]);*/
if(rcount[0]>=RCMAX/**CPMAX_-bdrnum*/) return 1;
```

```
else return 0;
}
else{
/*printf(" cag_r:%ld\n",rcount[0]);*/
return 0;
}
}/** cag_r_6 **/
#else
int cag_r_4(int nax_,int nay_,int color_)
{
int i;
int flag_[CPMAX],flag_pp_[CPMAX];
int nax[CPMAX],nay[CPMAX];
int ca,acolor[CPMAX];
int cp,algo,ssize;
long fpos[CPMAX];
double val,angle;

ssize=sizeof(s);
cp=1;

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

if(!zeroFill){
     if(tmp4==2) ca=16;             /* C+0 */
else if(tmp4==3) ca=-1;             /* S_ */
else if(tmp4==4) ca=16;             /* 0+S */
else if(tmp4==5) ca=-1;             /* C_ */
else             ca=/*15*/16;
}
else             ca=zg;             /* S_ */

nax[0]=nax_;nay[0]=nay_;
acolor[0]=color_;

if(!zeroFill){
if(pixel_[nax[0]][nay[0]]!=ca) return 1;
}

if(acolor[0]!=ca){
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]);
rcount[i]++;

     if(tmp4==1){                     /* i=0 */
bdr[bdrnum].cx=nx[i];bdr[bdrnum].cy=ny[i];
bdrnum++;
}
else if(tmp4==2) write_neighbor(3,acolor[i]);
}/**if(flag_[i])**/

i++;break;
}/**while(1)**/
}/**if(acolor[])**/
else cp=0;

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

while(cp){
if(refill==0) {refill=2;break;}

i=0;
while(1){
/*use_subroop();*/                        /* kbhit_() */
/*if(refill==0) break;*/

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

nxp=nx[i]+1;nyp=ny[i]+1;nxm=nx[i]-1;nym=ny[i]-1;
nxp_c=connect_nxpm(nxp);nxm_c=connect_nxpm(nxm);
nyp_c=connect_nypm(nyp);nym_c=connect_nypm(nym);
c1=getpixel_(nxp_c,ny[i]);c2=getpixel_(nx[i],nyp_c);c3=getpixel_(nxm_c,ny[i]);
c4=getpixel_(nx[i],nym_c);
c5=getpixel_(nxp_c,nyp_c);c7=getpixel_(nxm_c,nym_c);
c6=getpixel_(nxm_c,nyp_c);c8=getpixel_(nxp_c,nym_c);

if((c1==ca)||(c2==ca)||(c3==ca)||(c4==ca)){
s.xx=nx[i];s.yy=ny[i];fwrite_mem(i);

std_x=nx[i];std_y=ny[i];              /* S */

if(tmp4==2) write_neighbor(2,acolor[i]);
```

```
  /* CW (no phase) */
     if(c1==ca) {nx[i]=nxp_c;ny[i]=std_y;}
else if(c2==ca) {nx[i]=std_x;ny[i]=nyp_c;}
else if(c3==ca) {nx[i]=nxm_c;ny[i]=std_y;}
else if(c4==ca) {nx[i]=std_x;ny[i]=nym_c;}

/*else if(c5==ca) {nx[i]=nxp_c;ny[i]=nyp_c;}
else if(c6==ca) {nx[i]=nxm_c;ny[i]=nyp_c;}
else if(c7==ca) {nx[i]=nxm_c;ny[i]=nym_c;}
else if(c8==ca) {nx[i]=nxp_c;ny[i]=nym_c;}*/

/*jump(i,nx[i],ny[i]);*/                /* modification of b,S */

putpixel(nx[i],ny[i],acolor[i]);
rcount[i]++;

     if(tmp4==1){                    /* i=0 */
bdr[bdrnum].cx=nx[i];bdr[bdrnum].cy=ny[i];
bdrnum++;
}
else if(tmp4==2) write_neighbor(3,acolor[i]);

flag_pp_[i]=1;
}/**if(c1,c2,c3,c4)**/
else{
flag_pp_[i]=0;

if(ftell_mem(i)==0) {flag_[i]=0;cp--;if(cp==0) break;}  /* 2pie/3 */ /* pie */
fread_mem(i);
nx[i]=s.xx;ny[i]=s.yy;
}/**else(c1,c2,c3,c4)**/
}/**if(flag_[i])**/

i++;break;
}/**while(1)**/

if(flag_pp_[0]==1){
}

if(flag_pp_[0]==0){
}
}/**while(cp)**/

if((c_trans==1 && !SPmode)||(zeroFill)){
/*printf(" %ld\n",rcount[0]);*/
```

```
if(rcount[0]>=RCMAX/**CPMAX_-bdrnum*/) return 1;
else return 0;
}
else{
/*printf(" cag_r_4:%ld\n",rcount[0]);*/
return 0;
}
}/** cag_r_4 **/


int cag_r_8(int nax_,int nay_,int color_)
{
int i;
int flag_[CPMAX],flag_pp_[CPMAX];
int nax[CPMAX],nay[CPMAX];
int ca,acolor[CPMAX];
int cp,algo,ssize;
long fpos[CPMAX];
double val,angle;

/*if(color_<0) return 1;*/

ssize=sizeof(s);
cp=1;

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

if(!zeroFill){
     if(tmp4==2) ca=16;             /* C+0 */
else if(tmp4==3) ca=-1;             /* S_ */
else if(tmp4==4) ca=16;             /* 0+S */
else if(tmp4==5) ca=-1;             /* C_ */
else             ca=/*15*/16;
}
else             ca=zg;             /* S_ */

nax[0]=nax_;nay[0]=nay_;
acolor[0]=color_;

if(!zeroFill){
if(pixel_[nax[0]][nay[0]]!=ca) return 1;
}
```

```
if(acolor[0]!=ca){
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]);
rcount[i]++;

     if(tmp4==1){                  /* i=0 */
bdr[bdrnum].cx=nx[i];bdr[bdrnum].cy=ny[i];
bdrnum++;
}
else if(tmp4==2) write_neighbor(3,acolor[i]);
}/**if(flag_[i])**/

i++;break;
}/**while(1)**/
}/**if(acolor[])**/
else cp=0;

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

while(cp){
if(refill==0) {refill=2;break;}

i=0;
while(1){
/*use_subroop();*/                     /* kbhit_() */
/*if(refill==0) break;*/

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

nxp=nx[i]+1;nyp=ny[i]+1;nxm=nx[i]-1;nym=ny[i]-1;
nxp_c=connect_nxpm(nxp);nxm_c=connect_nxpm(nxm);
nyp_c=connect_nypm(nyp);nym_c=connect_nypm(nym);
c1=getpixel_(nxp_c,ny[i]);c2=getpixel_(nx[i],nyp_c);c3=getpixel_(nxm_c,ny[i]);
c4=getpixel_(nx[i],nym_c);
c5=getpixel_(nxp_c,nyp_c);c7=getpixel_(nxm_c,nym_c);
c6=getpixel_(nxm_c,nyp_c);c8=getpixel_(nxp_c,nym_c);

if((c1==ca)||(c2==ca)||(c3==ca)||(c4==ca)||(c5==ca)||(c7==ca)||(c6==ca)||(c8==ca)){
s.xx=nx[i];s.yy=ny[i];fwrite_mem(i);
std_x=nx[i];std_y=ny[i];            /* S */

if(tmp4==2) write_neighbor(2,acolor[i]);

  /* CW (no phase) */
     if(c1==ca) {nx[i]=nxp_c;ny[i]=std_y;}
else if(c2==ca) {nx[i]=std_x;ny[i]=nyp_c;}
else if(c3==ca) {nx[i]=nxm_c;ny[i]=std_y;}
else if(c4==ca) {nx[i]=std_x;ny[i]=nym_c;}

else if(c5==ca) {nx[i]=nxp_c;ny[i]=nyp_c;}
else if(c6==ca) {nx[i]=nxm_c;ny[i]=nyp_c;}
else if(c7==ca) {nx[i]=nxm_c;ny[i]=nym_c;}
else if(c8==ca) {nx[i]=nxp_c;ny[i]=nym_c;}

/*jump(i,nx[i],ny[i]);*/                /* modification of b,S */

putpixel(nx[i],ny[i],acolor[i]);
rcount[i]++;

     if(tmp4==1){                  /* i=0 */
bdr[bdrnum].cx=nx[i];bdr[bdrnum].cy=ny[i];
bdrnum++;
}
else if(tmp4==2) write_neighbor(3,acolor[i]);

flag_pp_[i]=1;
}/**if(c1,c2,c3,c4)**/
else{
flag_pp_[i]=0;

if(ftell_mem(i)==0) {flag_[i]=0;cp--;if(cp==0) break;}  /* 2pie/3 */ /* pie */
fread_mem(i);
nx[i]=s.xx;ny[i]=s.yy;
}/**else(c1,c2,c3,c4)**/
}/**if(flag_[i])**/

i++;break;
}/**while(1)**/

if(flag_pp_[0]==1){
}

if(flag_pp_[0]==0){
}
```

```
}/**while(cp)**/

if((c_trans==1 && !SPmode)||(zeroFill)){
/*printf(" %ld\n",rcount[0]);*/
if(rcount[0]>=RCMAX/**CPMAX_-bdrnum*/) return 1;
else return 0;
}
else{
/*printf(" cag_r_8:%ld\n",rcount[0]);*/
return 0;
}
}/** cag_r_8 **/
#endif
```