Cellular Automaton Graphics(13)

Kikuchi Morio

Abstract :

In body-centred cubic lattice and face-centred cubic lattice which are discrete coordinate system with central part, we do a tiling.

1. Leading rotation

If dimension number is over 2 and PBC is 0, we must give attention to leading rotation in symmetry. Leading rotation is a plane algorithm which becomes the centre in a painting algorithm. In 3 dimensions or higher, en_xy() is leading rotation without some programs. There is a program in which it is not. We show the mechanism of a 3-dimensional painting algorithm simplifying it.

```
・en_xy()+en_yz()+en_zx()          /* xy/en_xy() */
・en_xy()+en_zx()+en_yz()          /* xy/en_xy() */
```

/en_xy() expresses that en_xy() is leading rotation. Both are used according to the value of "painting point number"/2. In order to make a picture symmetric on a line which is vertical to xy plane, we adjust the way of a rotaion according to painting point number in en_yz() and en_zx(). We call the painting algorithm xy. We can use the following painting algorithms too.

```
・en_yz()+en_zx()+en_xy()          /* yz/en_xy() */
・en_zx()+en_yz()+en_xy()          /* yz/en_xy() */

・en_zx()+en_yz()+en_xy()          /* zx/en_xy() */
・en_yz()+en_zx()+en_xy()          /* zx/en_xy() */
```

In yz and zx, in order to make a picture symmetric on a line which is vertical to xy plane, we arrange en_xy() at the end and we adjust the way of a rotaion according to painting point number in en_yz() and en_zx().

In 4 dimensions, it becomes like the following:

```
・en_xy()+en_yz()+en_zx() +en_xu()+en_yu()+en_zu()  /* xy/en_xy() */
・en_xy()+en_zx()+en_yz() +en_yu()+en_xu()+en_zu()  /* xy/en_xy() */

・en_yz()+en_zx()+en_xy() +en_xu()+en_yu()+en_zu()  /* yz/en_xy() */
・en_zx()+en_yz()+en_xy() +en_yu()+en_xu()+en_zu()  /* yz/en_xy() */

・en_zx()+en_yz()+en_xy() +en_xu()+en_yu()+en_zu()  /* zx/en_xy() */
・en_yz()+en_zx()+en_xy() +en_yu()+en_xu()+en_zu()  /* zx/en_xy() */
```

In order to make a picture symmetric on a line which is vertical to xy plane, we adjust the way of a rotaion according to painting point number in en_xu() and en_yu().

If dimension number is over 4, the arrangement from the seventh item is automatical.

```
・                          +en_xu()+en_yu()+en_zu()
                           +en_xv()+en_yv()+en_zv()+en_uv()
                           + ...
                           +en_yu()+en_xu()+en_zu()
                           +en_yv()+en_xv()+en_zv()+en_uv()
                           + ...
```

In order to make a picture symmetric on a line which is vertical to xy plane, we adjust the way of a rotaion according to painting point number in en_x?() and en_y?().

If en_xy() is leading rotation, in x-y-? coordinate system, a picture is symmetric on a line which is vertical to xy plane. For example, in 10d.4, we choose 6 dimensions and we make Xlib 1.

```
#define DMS 6
#define Xlib 1
```

We arrange seed points on two planes in the direction of z for example z=0+1 and z=RESO-1-1 4 points each. In void field(void), we do the modification like the following:

```
#if /*PBC*/1
/* all */
```

In int putpixel(...), we do the modification like the following:

```
if(sn_==-1){
if(u==ut && v==vt && z==zt && r==0 && s==0 && t==0 && o==0){
if(x>=fr && x<=to && y>=fr && y<=to && w>=fr && w<=to)
pixel_3d(x*PS,y*PS,w*PS,pcolor);
}
}
```

Because it is 6 dimensions, we make the variables r, s, t, o which are not connected with 6 dimensions 0. If we do the above modifications, we get a symmetric picture on x, y, w.

If en_yz() is leading rotation, in y-z-? coordinate system, we get a picture which is symmetric on a line which is vertical to yz plane and if en_zx() is leading rotation, in z-x-? coordinate system, we get a picture which is symmetric on a line which is vertical to zx plane. We try making en_yz() and en_zx() leading rotation. In order to avoid disorder, we make 3d.8 the basis and minimize the number of painting algorithms.
If en_yz() is leading rotation

```
・en_yz()+en_zx()+en_xy()          /* yz/en_yz() */
・en_yz()+en_xy()+en_zx()          /* yz/en_yz() */
```

We arrange seed points on two planes in the direction of x for example x=0+1 and x=RESO-1-1 4 points each.
If en_zx() is leading rotation

```
・en_zx()+en_xy()+en_yz()          /* zx/en_zx() */
・en_zx()+en_yz()+en_xy()          /* zx/en_zx() */
```

We arrange seed points on two planes in the direction of y for example y=0+1 and y=RESO-1-1 4 points each.

We may make the number of seed points 4 × 2 of one direction. If we consider the space symmetry, 4 × 2 × 3 of three directions is possible too.

```
#define DMS 3
#define CPMAX 24
#define VGACOLORS 24
```

We arrange the 2nd painting points on the plane on which the seed points are placed at first. In this case, we paint the pixels which are penetrated by four lines which join two vertexes of the region in advance. We call the work closing of pixel.

```
#define cpwidth 2
#define delta_x 2
#define RESO (cpwidth*(2*delta_x+1))
```

In the above case, RESO is 10 and the number of pixels of the region becomes 1000. Because the number of closed pixels is 10 × 4, the number of pixels which are painted becomes 960. Note that 960 is divisible by CPMAX. 3d.10 results from the above work. The number of axes of symmetry of a picture becomes 3. Figure 1 is a painting example of 3d.10.

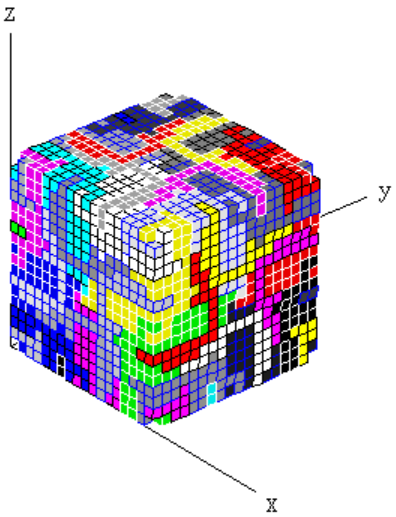


Figure 1

In 3d.10, the number of seed points is 4 × 2 per one direction. 2 × 2 is possible too. In 3d.9, 2 × 2 points are arranged in the direction of z. If we see the region from the direction of z, the two lines which join two seed points look like x. Because it is one direction, closing of pixel is not needed. In 3d.8 in which the number of painting points is eight, if we make the painting point number 2, 3, 4, 5 inactive, the program is got.

2. Making of equal volume 14-faced polyhedron

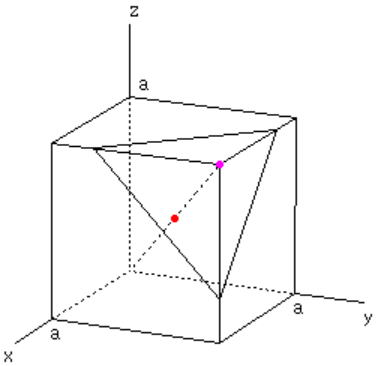


Figure 2

We think about 3-dimensional space which is tiled by cube namely regular hexahedron. We cut off the vertex and 3/4 of the side like Figure 2. Assuming the length of the side to be a, the volume of the triangular pyramid is

$$(3a/4)^3/6 = 27a^3/(6 \cdot 64)$$

We do the cut on all the vertexes. After the cuts, we get a polyhedron which has 6 squares and 8 hexagons. The overlap part of the two triangular pyramids is two triangular pyramids. The volume is

$$2 \cdot (1/6)\sqrt{(\sqrt{2}a/4)^2 - (0.5a/2)^2}\sqrt{(\sqrt{2}a/4)^2 - (0.5a/2)^2}(0.5a/2) = a^3/(3 \cdot 64)$$

The volume after the cuts is

$$a^3 - 8{\cdot}27a^3/(6 \cdot 64) + 12{\cdot}a^3/(3 \cdot 64) = a^3/2$$

By the way, If we make 8 heptahedrons which are the same from the eliminated 8 polyhedrons and put them together, this is a polyhedron which has 6 squares and 8 hexagons too. That is, we got two 14-faced polyhedrons which are the same by the cuts and handicraft. If we do the cuts and handicraft imaginarily, it is obvious that the space is tiled by 14-faced polyhedron. Besides, the two centres of the two 14-faced polyhedrons are the two points on the diagonal which are shown in Figure 2. Therefore, tiling by 14-faced polyhedron corresponds to body-centred cubic lattice.

3. Body-centred cubic lattice

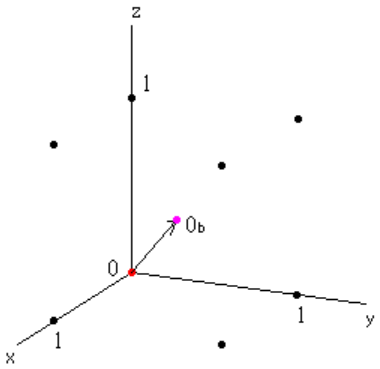


Figure 3

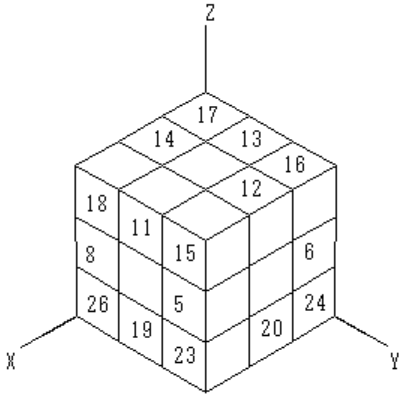
In body-centred cubic lattice, we use two coordinate systems K、K_b. K_b is got by a parallel movement of which amount of axial movement of K is 0.5 like Figure 3 and it expresses body-centre. O_b is the origin of K_b. The coordinate system number is 0, 1 respectively and we impose periodic boundary condition on them.

In movement between them, assuming ΔX etc to be the coordinate difference from the centre of a neighborhood shown in Figure 4, we use the following rule:

$\Delta : \Delta X,\ \Delta Y,\ \Delta Z$

$\mathbf{r}_b \to \mathbf{r}$: if $\Delta < 0$, invariable

$\mathbf{r} \to \mathbf{r}_b$: if $\Delta > 0$, invariable




```
15:ΔX=1, ΔY=1, ΔZ=1      23:ΔX=1, ΔY=1, ΔZ=-1
16:ΔX=-1, ΔY=1, ΔZ=1     24:ΔX=-1, ΔY=1, ΔZ=-1
17:ΔX=-1, ΔY=-1, ΔZ=1    25:ΔX=-1, ΔY=-1, ΔZ=-1
18:ΔX=1, ΔY=-1, ΔZ=1     26:ΔX=1, ΔY=-1, ΔZ=-1

11:ΔX=1, ΔY=0, ΔZ=1
19:ΔX=1, ΔY=0, ΔZ=-1
21:ΔX=-1, ΔY=0, ΔZ=-1
13:ΔX=-1, ΔY=0, ΔZ=1

12:ΔX=0, ΔY=1, ΔZ=1
20:ΔX=0, ΔY=1, ΔZ=-1
22:ΔX=0, ΔY=-1, ΔZ=-1
14:ΔX=0, ΔY=-1, ΔZ=1

5:ΔX=1, ΔY=1, ΔZ=0
6:ΔX=-1, ΔY=1, ΔZ=0
7:ΔX=-1, ΔY=-1, ΔZ=0
8:ΔX=1, ΔY=-1, ΔZ=0
```

Figure 4

In the above movement, we use the following neighborhoods:

・15, 16, 17, 18, 23, 24, 25, 26

We use both rotation type and neighborhood description type in painting algorithm.

・en_xy()+en_yz()+en_zx()　　/* xy/en_xy() */

・15, 16, 17, 18, 23, 24, 25, 26

・25, 26, 23, 24, 17, 18, 15, 16

On neighborhood description, we prepare the two sets which are in a recursive swap. There is a relation that CPMAX=pow(cpwidth,3)*pow(2,n)(n=0,1). When n=1, we place seed points in both the coordinate systems and we use the two sets switching them. If we use only one set, the number of painting patterns becomes two.

4. Face-centred cubic lattice

I do not have a knowledge of tiling by one kind of pixel which corresponds to face-centred cubic lattice. However, here, we do a painting noticing not pixel but a set of coordinate systems in relation to body-centred cubic lattice.

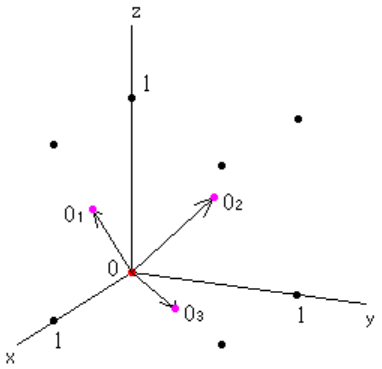


Figure 5

In face-centred cubic lattice, we use four coordinate systems K, K_1, K_2, K_3. K_1, K_2, K_3 are got by a parallel movement of which amount of axial movement of K is 0.5 like Figure 5 and they express face-centre. O_1, O_2, O_3 are the origins of K_1, K_2, K_3 respectively. The coordinate system number is 0, 1, 2, 3 respectively and we impose periodic boundary condition on them.

In movement between two coordinate systems, assuming ΔX etc to be the coordinate difference from the centre of a neighborhood shown in Figure 4, we use the following rule:

Δ:ΔX, ΔZ

$\mathbf{r}_1 \to \mathbf{r}$: if $\Delta < 0$, invariable

$\mathbf{r} \to \mathbf{r}_1$: if $\Delta > 0$, invariable

Δ:ΔY, ΔZ

$\mathbf{r}_2 \to \mathbf{r}$: if $\Delta < 0$, invariable

$\mathbf{r} \to \mathbf{r}_2$: if $\Delta > 0$, invariable

Δ:ΔX, ΔY

$\mathbf{r}_3 \to \mathbf{r}$: if $\Delta < 0$, invariable

$\mathbf{r} \to \mathbf{r}_3$: if $\Delta > 0$, invariable

$\mathbf{r}_1 \to \mathbf{r}_2$: if $\Delta X < 0$, invariable; if $\Delta Y > 0$, invariable

$\mathbf{r}_2 \to \mathbf{r}_1$: if $\Delta X > 0$, invariable; if $\Delta Y < 0$, invariable

$\mathbf{r}_2 \to \mathbf{r}_3$: if $\Delta X > 0$, invariable; if $\Delta Z < 0$, invariable

$\mathbf{r}_3 \to \mathbf{r}_2$: if $\Delta X < 0$, invariable; if $\Delta Z > 0$, invariable

$\mathbf{r}_3 \to \mathbf{r}_1$: if $\Delta Y < 0$, invariable; if $\Delta Z > 0$, invariable

$\mathbf{r}_1 \to \mathbf{r}_3$: if $\Delta Y > 0$, invariable; if $\Delta Z < 0$, invariable

In the above movement, we use the following neighborhoods:

$\mathbf{r}_1, \mathbf{r}$: 11, 19, 21, 13

$\mathbf{r}_2, \mathbf{r}$: 12, 14, 22, 20

$\mathbf{r}_3, \mathbf{r}$: 5, 6, 7, 8

$\mathbf{r}_1, \mathbf{r}_2$: 5, 6, 7, 8

$\mathbf{r}_2, \mathbf{r}_3$: 11, 19, 21, 13

$\mathbf{r}_3, \mathbf{r}_1$: 12, 14, 22, 20

We use both rotation type and neighborhood description type in painting algorithm.

・en_xy()+en_yz()+en_zx()　　/* xy/en_xy() */

・11, 19, 21, 13
・12, 14, 22, 20
・5, 6, 7, 8

・21, 13, 11, 19
・22, 20, 12, 14

・7, 8, 5, 6

On neighborhood description, we prepare the two sets which are in a swap. There is a relation that CPMAX=pow(cpwidth,3)*pow(2,n)(n=0,1,2). When n=1, we place seed points in two coordinate systems and we use the two sets switching them. If we use only one set, the number of painting patterns becomes two.

5. Program

We use the following programs:

・3d.12 : body-centred cubic lattice
・3d.15 : face-centred cubic lattice

We choose cpwidth and CPMAX.

```
#define cpwidth 2
#define CPMAX 16
```

If CPMAX is too large, there is a probability that colors do not appear. In this case, enlarge VGACOLORS. If colors do not appear nevertheless, modify palette(initpalette()).
We can change the size of the array with delta_x.

```
#define delta_x 2
#define RESO (cpwidth*(2*delta_x+1))
```

If _3d14 is 0 or -2, we use all the coordinate systems namely coordinate sytem number 0, 1, 2, 3. If _3d14 is 1, we use 0 and the following K_:

```
#define _3d14 1
#define K_ 3                          /* >0:auto, 0:manual */
```

We call seed points which are arranged in one coordinate system set too. If cpwidth is 2, it is 8 seed points and if cpwidth is 3, it is 27 seed points. In 2 dimensions, we could choose parallelism or antiparallelism in the direction of the 2nd painting point. Therefore, in a set, we can choose parallelism or antiparallelism in the direction of the 2nd painting point. However, in this case, if we want to make it antiparallel, we must make cpwidth an even number, and so, here, for the sake of ease, in a set, we adopt parallelism in the direction of the 2nd painting point.
When the number of sets is 2, if we want to make the number of painting patterns two, make _3d14 -2 or match the following -- or ++ to the other.

```
if(PBC || CPMAX==1){
if(CPMAX==CPMAXh*2 && /*i<CPMAXh*/(i/CPMAXh)%2==0) nax[i]--;
else                              nax[i]++;
}
```

6. Painting example

We show a painting example of face-centred cubic lattice.

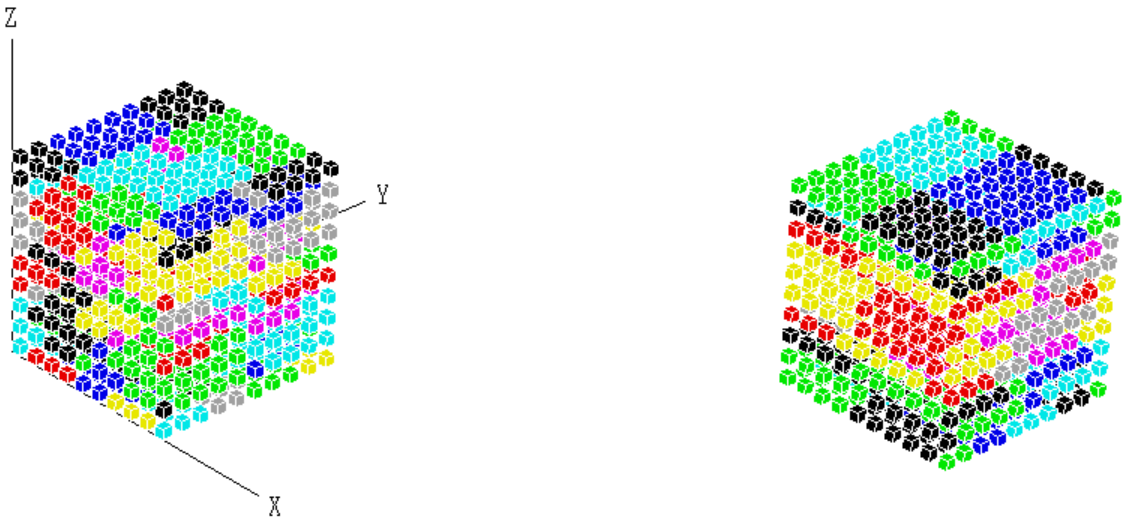


Figure 6

Figure 6 is a picture got by the following parameters:

```
#define cpwidth 2
#define CPMAX 8
#define _3d14 1
#define K_ 3                          /* >0:auto, 0:manual */
#define delta_x 2
```

CPMAX is 8 and it is one set. In the side face too, the same color pattern appears at intervals of four pixel rows.

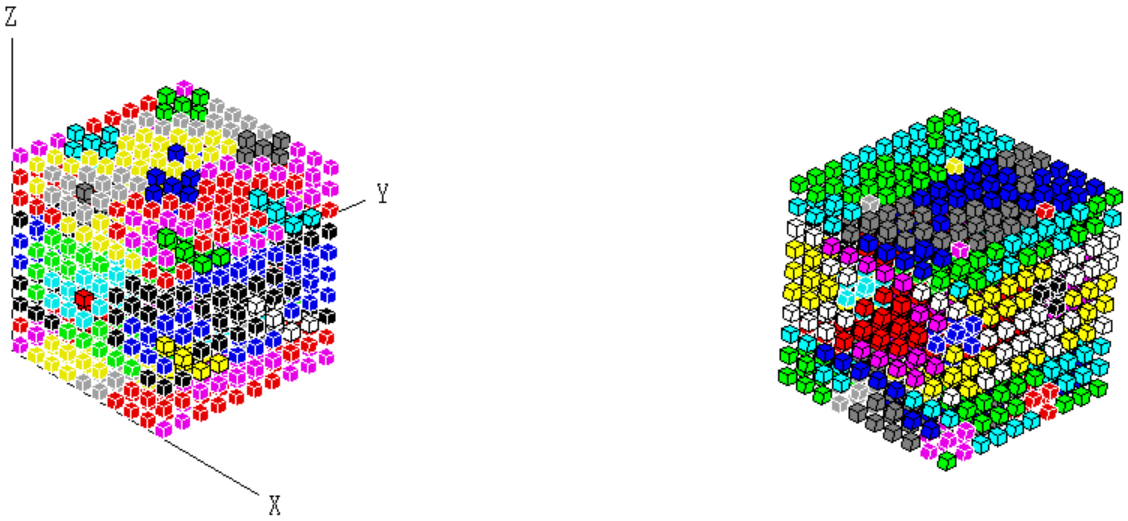


Figure 7

Figure 7 is a picture got by the following parameters:

```
#define cpwidth 2
#define CPMAX 16
#define _3d14 1
#define K_ 3                          /* >0:auto, 0:manual */
#define delta_x 2
```

CPMAX is 16 and it is two sets. We place the 1st set in K and the 2nd set in K_. The border color of pixel is white and black respectively.

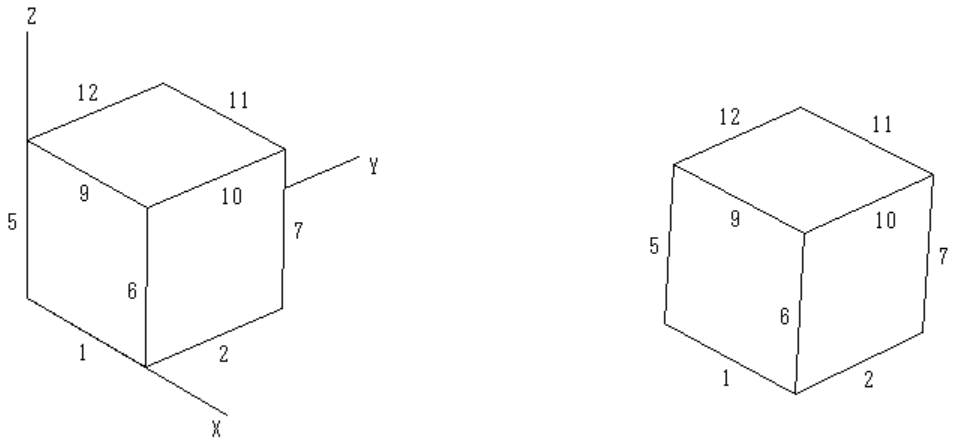


Figure 8

The number in Figure 8 is a side number. The same color pattern appears on the two corresponding sides of the two pictures antiparallel.

- $1 \Longleftrightarrow 11$
- $2 \Longleftrightarrow 12$
- $5 \Longleftrightarrow 7$
- $7 \Longleftrightarrow 5$
- $11 \Longleftrightarrow 1$
- $12 \Longleftrightarrow 2$

****

セルラーオートマトングラフィクス (13)

菊池盛雄

アブストラクト：
有心型離散座標系である体心立方格子、面心立方格子において塗りつぶしを行います

1. 基礎回転

3 次元以上では PBC が 0 の場合、対称性については基礎回転に留意しなければなりません。基礎回転とは塗りつぶしアルゴリズムの中で中心となる平面アルゴリズムを指します。3 次元以上では一部を除いて en_xy() が基礎回転です。基礎回転がない場合もあります。3 次元の場合について簡略化して示します。

```
・en_xy()+en_yz()+en_zx()          /* xy/en_xy() */
・en_xy()+en_zx()+en_yz()          /* xy/en_xy() */
```

/en_xy() は en_xy() が基礎回転であることを表します。両者は塗点番号/2 の値に応じて使い分けます。画像を xy 平面に垂直な直線に関して対称にするために、en_yz()、en_zx() に関しては塗点番号に応じて回転の向きを調整します。この塗りつぶしアルゴリズムを xy と称します。以下の塗りつぶしアルゴリズムを用いてもかまいません。

```
・en_yz()+en_zx()+en_xy()          /* yz/en_xy() */
・en_zx()+en_yz()+en_xy()          /* yz/en_xy() */

・en_zx()+en_yz()+en_xy()          /* zx/en_xy() */
・en_yz()+en_zx()+en_xy()          /* zx/en_xy() */
\begin{verbatim}
\ \\
```

yz、zx においては、画像を xy 平面に垂直な直線に関して対称にするために、en_xy() を末尾に配置し、en_yz()、en_zx() に関しては塗点番号に応じて回転の向きを調整します。\\

4 次元では以下のような配置になります。\\

```
\begin{verbatim}
・en_xy()+en_yz()+en_zx() +en_xu()+en_yu()+en_zu()  /* xy/en_xy() */
・en_xy()+en_zx()+en_yz() +en_yu()+en_xu()+en_zu()  /* xy/en_xy() */

・en_yz()+en_zx()+en_xy() +en_xu()+en_yu()+en_zu()  /* yz/en_xy() */
・en_zx()+en_yz()+en_xy() +en_yu()+en_xu()+en_zu()  /* yz/en_xy() */

・en_zx()+en_yz()+en_xy() +en_xu()+en_yu()+en_zu()  /* zx/en_xy() */
・en_yz()+en_zx()+en_xy() +en_yu()+en_xu()+en_zu()  /* zx/en_xy() */
```

画像を xy 平面に垂直な直線に関して対称にするために、en_xu()、en_yu() に関しては塗点番号に応じて回転の向きを調整します。5 次元以上では第 7 項からは機械的に配置します。

```
・                          +en_xu()+en_yu()+en_zu()
                            +en_xv()+en_yv()+en_zv()+en_uv()
                            + ...
                            +en_yu()+en_xu()+en_zu()
                            +en_yv()+en_xv()+en_zv()+en_uv()
                            + ...
```

画像を xy 平面に垂直な直線に関して対称にするために、en_x?()、en_y?() に関しては塗点番号に応じて回転の向きを調整します。en_xy() が基礎回転であれば x-y-?座標系において画像が xy 平面に垂直な直線に関して対称になります。例えば 10d.4 において 6 次元とし、Xlib を 1 とします。

```
#define DMS 6
#define Xlib 1
```

シードポイントは z 方向の 2 平面例えば z=0+1、z=RESO-1-1 に各々 4 個ずつ配置します。void field(void) において以下のような修正を行います。

```
#if /*PBC*/1
/* all */
```

int putpixel(...) において以下のような修正を行います。

```
if(sn_==-1){
if(u==ut && v==vt && z==zt && r==0 && s==0 && t==0 && o==0){
if(x>=fr && x<=to && y>=fr && y<=to && w>=fr && w<=to)
pixel_3d(x*PS,y*PS,w*PS,pcolor);
}
}
```

6 次元ですから 6 次元オーバーの変数である r、s、t、o は 0 とします。このようにすれば x、y、w に関する対称な画像が得られます。
　もし en_yz() を基礎回転に選べば、y-z-?座標系において yz 平面に垂直な直線に関して対称な画像が、en_zx() を基礎回転に選べば、z-x-?座標系において zx 平面に垂直な直線に関して対称な画像が得られます。en_yz()、en_zx() を基礎回転にしてみます。混乱を避けるために 3d.8 を元とし、塗りつぶしアルゴリズムの数は最少に留めます。
en_yz() が基礎回転であれば

```
・en_yz()+en_zx()+en_xy()          /* yz/en_yz() */
・en_yz()+en_xy()+en_zx()          /* yz/en_yz() */
```

シードポイントは x 方向の 2 平面例えば x=0+1、x=RESO-1-1 に各々 4 個ずつ配置します。
en_zx() が基礎回転であれば

```
・en_zx()+en_xy()+en_yz()          /* zx/en_zx() */
・en_zx()+en_yz()+en_xy()          /* zx/en_zx() */
```

シードポイントは y 方向の 2 平面例えば y=0+1、y=RESO-1-1 に各々 4 個ずつ配置します。
　シードポイントの数は単方向の 4 × 2 個としてもかまいませんが、空間的な対称性を考慮すれば 3 方向の 4 × 2 × 3 個も可能です。

```
#define DMS 3
#define CPMAX 24
#define VGACOLORS 24
```

第二塗点は最初はシードポイントのある平面上に配置します。この場合、領域の頂点を結ぶ線に突き抜かれるピクセルをあらかじめ塗りつぶしておきます。この作業を済化と称します。

```
#define cpwidth 2
#define delta_x 2
#define RESO (cpwidth*(2*delta_x+1))
```

上の場合、RESO は 10 であり、領域のピクセル数は 1000 となります。済化ピクセルの数は 10 × 4 であるので、塗りつぶされるピクセル数は 960 となります。960 が CPMAX で割り切れることに注意してください。結果は 3d.10 となります。画像の対称軸は 3 本となります。図 1 は 3d.10 の描画例です。

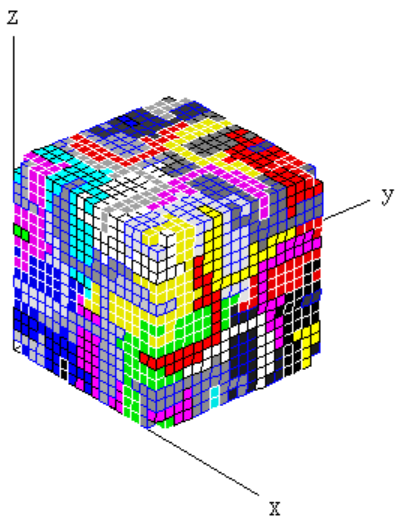


図 1

3d.10 においてはシードポイントの数は単方向当たり 4 × 2 個としています。シードポイントの数は単方向当たり 2 × 2 個も可能です。3d.9 では 2 × 2 個を z 方向に配置しています。z 方向から見るとシードポイントを結ぶ二つの直線は直交するように見えます。単方向なので済化は必要ありません。3d.8 において、塗点番号 0 から 7 の内、2 から 5 を inactive にすればこのプログラムが得られます。

2. 等積 14 面体の作成

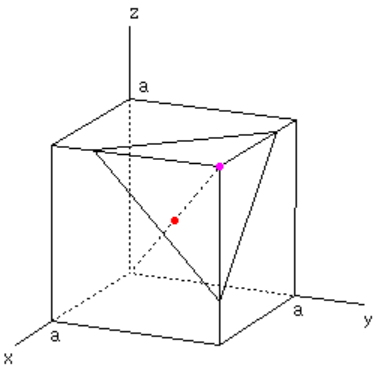


図 2

立方体すなわち正六面体で敷きつめられた三次元空間を考えます。正六面体の頂点と辺の 3/4 を図 2 のように切除します。一辺の長さを a とするとこの三角錐の体積は

$$(3a/4)^3/6 = 27a^3/(6\cdot 64)$$

この切除を全ての頂点に関して行います。正六面体は切除後正方形が 6 個、正六角形が 8 個の 14 面体となります。二つの三角錐の重複部分は二つの三角錐であり、その体積は

$$2\cdot(1/6)\sqrt{(\sqrt{2}a/4)^2-(0.5a/2)^2}\sqrt{(\sqrt{2}a/4)^2-(0.5a/2)^2}(0.5a/2) = a^3/(3\cdot 64)$$

切除後の体積は

$$a^3 - 8\cdot 27a^3/(6\cdot 64) + 12\cdot a^3/(3\cdot 64) = a^3/2$$

ところで、切除された八つの多面体から八つの同じ七面体を作ってこれらを組合わせると、これも正方形が 6 個、正六角形が 8 個の 14 面体です。つまり、切除と工作によって二つの同一の 14 面体が作成されたことになります。切除と七面体の作成・組合わせを仮想的に行えば三次元空間が 14 面体によって敷きつめられていることは明らかです。しかもこの二つの 14 面体の中心は図 2 の対角線上の丸で示される 2 点です。したがって、14 面体による敷きつめは体心立方格子と対応しています。

3. 体心立方格子

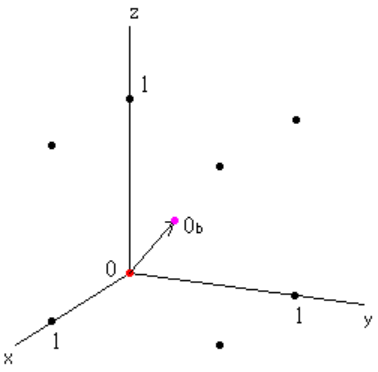


図 3

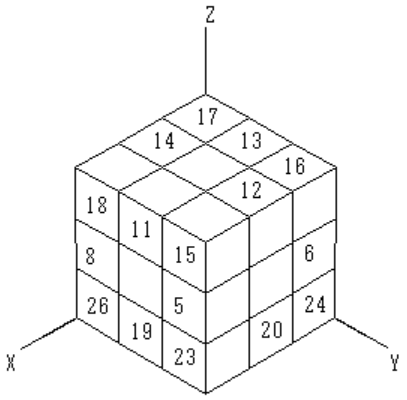
体心立方格子では二つの座標系 K、K_b を用います。K_b は K を軸方向の移動量 0.5 で図 3 のように平行移動させたものであり、体心を表します。O_b は K_b の原点です。座標系番号は各々 0、1 とし、周期的境界条件を課します。

両座標系間の移動においては ΔX 等を図 4 に示される近傍の変化量として以下のような規則を用います。

Δ:ΔX, ΔY, ΔZ

$\mathbf{r}_b \to \mathbf{r}$:$\Delta < 0$ なら不変

$\mathbf{r} \to \mathbf{r}_b$:$\Delta > 0$ なら不変



15:ΔX=1, ΔY=1, ΔZ=1　　23:ΔX=1, ΔY=1, ΔZ=-1
16:ΔX=-1, ΔY=1, ΔZ=1　　24:ΔX=-1, ΔY=1, ΔZ=-1
17:ΔX=-1, ΔY=-1, ΔZ=1　　25:ΔX=-1, ΔY=-1, ΔZ=-1
18:ΔX=1, ΔY=-1, ΔZ=1　　26:ΔX=1, ΔY=-1, ΔZ=-1

11:ΔX=1, ΔY=0, ΔZ=1
19:ΔX=1, ΔY=0, ΔZ=-1
21:ΔX=-1, ΔY=0, ΔZ=-1
13:ΔX=-1, ΔY=0, ΔZ=1

12:ΔX=0, ΔY=1, ΔZ=1
20:ΔX=0, ΔY=1, ΔZ=-1
22:ΔX=0, ΔY=-1, ΔZ=-1
14:ΔX=0, ΔY=-1, ΔZ=1

5:ΔX=1, ΔY=1, ΔZ=0
6:ΔX=-1, ΔY=1, ΔZ=0
7:ΔX=-1, ΔY=-1, ΔZ=0
8:ΔX=1, ΔY=-1, ΔZ=0

図 4

上記の移動においては以下の近傍を用います。

・15, 16, 17, 18, 23, 24, 25, 26

塗りつぶしアルゴリズムは回転型と近傍記述型を併用します。

・en_xy()+en_yz()+en_zx()　　/* xy/en_xy() */

・15, 16, 17, 18, 23, 24, 25, 26

・25, 26, 23, 24, 17, 18, 15, 16

近傍記述型に関しては再帰的スワップの関係にある 2 セットを用意します。CPMAX=pow(cpwidth,3)*pow(2,n)(n=0,1) という関係があり、n=1 ではシードポイントを両座標系に配置し、2 セットを切り替えて使用します。1 セットだけ使用すると描画パターンが二つになります。

4. 面心立方格子

面心立方格子と対応する一種類のピクセルによる敷き詰めについては筆者は知識を有していません。しかしながら、ここではピクセルではなく体心立方格子との関連で座標系の集合に着目して塗りつぶしを行います。

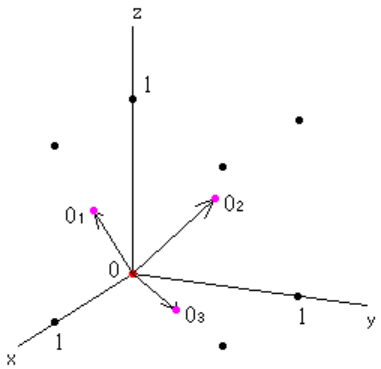


図 5

面心立方格子では四つの座標系 K、K_1、K_2、K_3 を用います。K_1、K_2、K_3 は K を軸方向の移動量 0.5 で図 5 のように平行移動させたものであり、面心を表します。O_1、O_2、O_3 は各々 K_1、K_2、K_3 の原点です。座標系番号は各々 0、1、2、3 とし、周期的境界条件を課します。

二座標系間の移動においては ΔX 等を図 4 に示される近傍の変化量として以下のような規則を用います。

Δ:ΔX, ΔZ

$\mathbf{r}_1 \to \mathbf{r}$:$\Delta < 0$ なら不変

$\mathbf{r} \to \mathbf{r}_1$:$\Delta > 0$ なら不変

Δ:ΔY, ΔZ

$\mathbf{r}_2 \to \mathbf{r}$:$\Delta < 0$ なら不変

$\mathbf{r} \to \mathbf{r}_2$:$\Delta > 0$ なら不変

Δ:ΔX, ΔY

$\mathbf{r}_3 \to \mathbf{r}$:$\Delta < 0$ なら不変

$\mathbf{r} \to \mathbf{r}_3$:$\Delta > 0$ なら不変

$\mathbf{r}_1 \to \mathbf{r}_2$:$\Delta X < 0$ なら不変。$\Delta Y > 0$ なら不変

$\mathbf{r}_2 \to \mathbf{r}_1$:$\Delta X > 0$ なら不変。$\Delta Y < 0$ なら不変

$\mathbf{r}_2 \to \mathbf{r}_3$:$\Delta X > 0$ なら不変。$\Delta Z < 0$ なら不変

$\mathbf{r}_3 \to \mathbf{r}_2$:$\Delta X < 0$ なら不変。$\Delta Z > 0$ なら不変

$\mathbf{r}_3 \to \mathbf{r}_1$:$\Delta Y < 0$ なら不変。$\Delta Z > 0$ なら不変

$\mathbf{r}_1 \to \mathbf{r}_3$:$\Delta Y > 0$ なら不変。$\Delta Z < 0$ なら不変

上記の移動においては以下の近傍を用います。

$\mathbf{r}_1, \mathbf{r}$: $11, 19, 21, 13$

$\mathbf{r}_2, \mathbf{r}$: $12, 14, 22, 20$

$\mathbf{r}_3, \mathbf{r}$: $5, 6, 7, 8$

$\mathbf{r}_1, \mathbf{r}_2$: $5, 6, 7, 8$

$\mathbf{r}_2, \mathbf{r}_3$: $11, 19, 21, 13$

$\mathbf{r}_3, \mathbf{r}_1$: $12, 14, 22, 20$

塗りつぶしアルゴリズムは回転型と近傍記述型を併用します。

・en_xy()+en_yz()+en_zx()　　/* xy/en_xy() */

・11, 19, 21, 13
・12, 14, 22, 20
・5, 6, 7, 8

・21, 13, 11, 19
・22, 20, 12, 14
・7, 8, 5, 6

近傍記述型に関してはスワップの関係にある 2 セットを用意します。CPMAX=pow(cpwidth,3)*pow(2,n)(n=0,1,2) という関係があり、n=1 ではシードポイントを二つの座標系に配置し、2 セットを切り替えて使用します。1 セットだけ使用すると描画パターンが二つになります。

5. プログラム

プログラムは 3d.12 と 3d.15 を用います。

・3d.12：体心立方格子
・3d.15：面心立方格子

cpwidth と CPMAX を選択します。

```
#define cpwidth 2
#define CPMAX 16
```

CPMAX を過大にした場合は色が表示されない場合があります。その場合は VGACOLORS を大きくしてください。それでも表示されないならパレット (initpalette()) を修正してください。
delta_x で配列の大きさを変えることができます。

```
#define delta_x 2
#define RESO (cpwidth*(2*delta_x+1))
```

_3d14 が 0 または-2 なら全ての座標系すなわち座標系番号 0、1、2、3 を使用します。_3d14 が 1 なら 0 と以下の K_ を使用します。

```
#define _3d14 1
#define K_ 3                          /* >0:auto, 0:manual */
```

一つの座標系に配置するシードポイントもセットと称します。cpwidth が 2 なら 1 セットは 8 シードポイント、cpwidth が 3 なら 1 セットは 27 シードポイントとなります。二次元においては第二塗点の方向を平行または反平行としていました。したがって、セット内において第二塗点の方向を平行または反平行とすることができます。しかし、この場合、反平行にするには cpwidth を偶数にしなければなりません。そこで、ここでは、簡単のため、セット内においては第二塗点の方向は平行にしておきます。
2 セットで描画パターンを二つにしたい場合は、_3d14 を-2 とするか以下の--または++を他方に合わせてください。

```
if(PBC || CPMAX==1){
if(CPMAX==CPMAXh*2 && /*i<CPMAXh*/(i/CPMAXh)%2==0) nax[i]--;
else                              nax[i]++;
}
```

6. 描画例
面心立方格子の場合の描画例を示します。

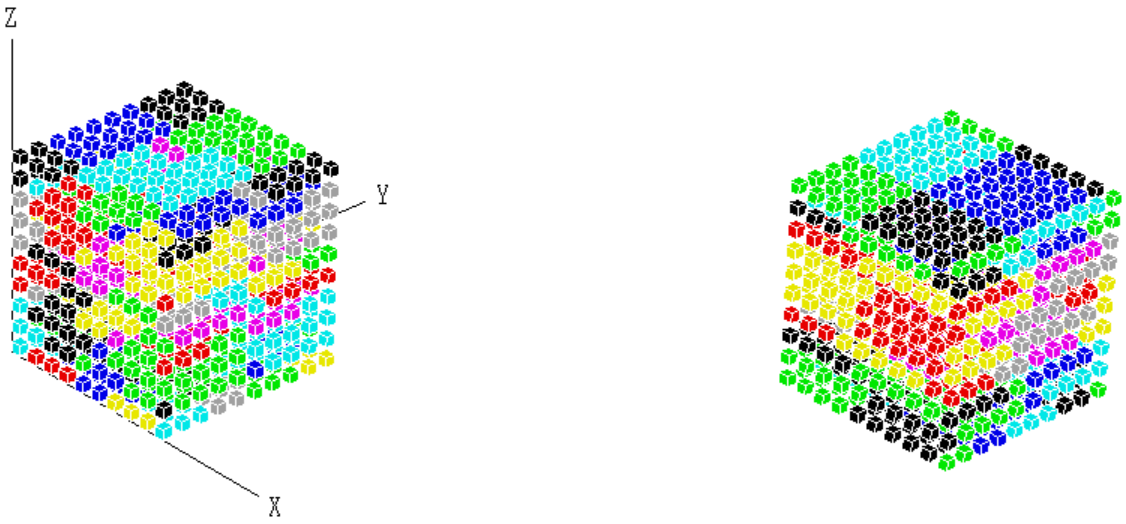


図 6

図 6 は以下のパラメーターで得られた画像です。

```
#define cpwidth 2
#define CPMAX 8
#define _3d14 1
#define K_ 3                          /* >0:auto, 0:manual */
#define delta_x 2
```

CPMAX が 8 であり、これは 1 セットです。側面においても同じ色パターンが四つのピクセル列を間において現れます。

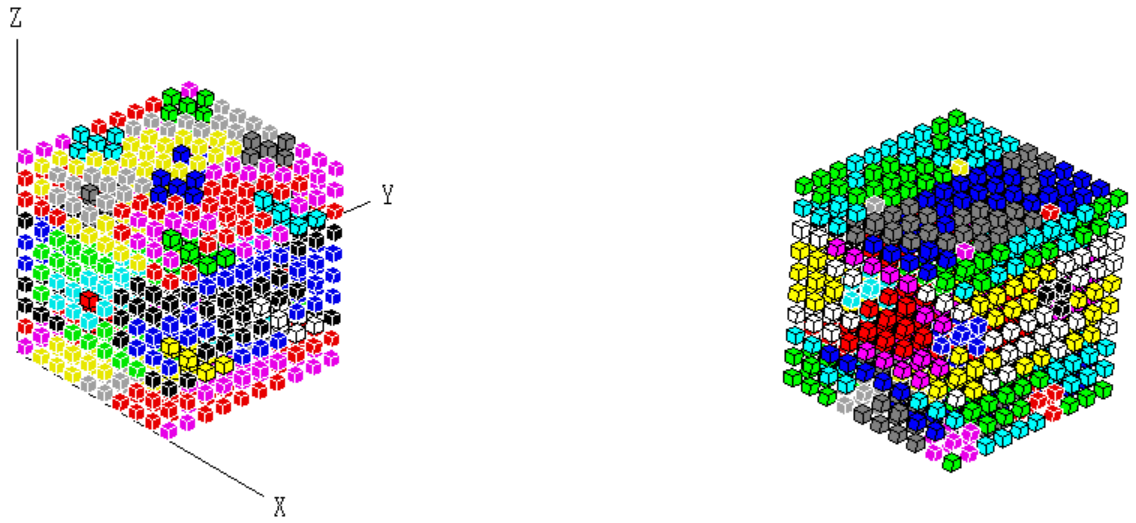


図 7

図 7 は以下のパラメーターで得られた画像です。

```
#define cpwidth 2
#define CPMAX 16
#define _3d14 1
#define K_ 3                          /* >0:auto, 0:manual */
#define delta_x 2
```

CPMAX が 16 であり、これは 2 セットです。K に第一セット、K_に第二セットを配置します。ピクセルの縁の色は各々白、黒です。

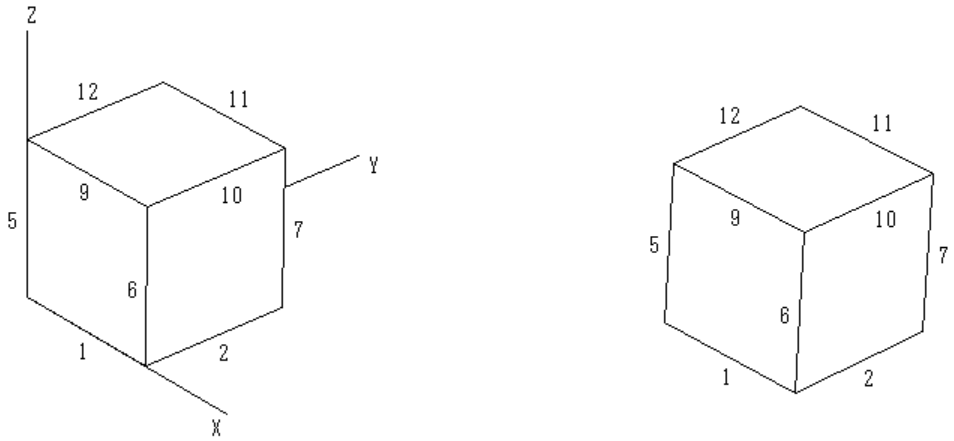


図 8

図 8 の数字は辺の番号です。左右の画像の対応する 2 辺に反平行的に同じ色パターンが現れます。

・1 ⟺ 11
・2 ⟺ 12
・5 ⟺ 7
・7 ⟺ 5
・11 ⟺ 1
・12 ⟺ 2

****

List 1:cag_13.c

```
/* 3d.15((SP4/p)*2)                                                    */
/* 2022 Kikuchi Morio                                                  */

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

#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>
#include <signal.h>

/* here **/
#define dbl double
#if WX==0
#define GKS GetKeyState
#define GKS_ GetKeyState
#define FOPEN fopen
#else
#define TRANS (65535./255)
#define XDSDY (asct+1)
#define FOPEN fopen64
#endif

/* here **/
#define PBC 1
#define DMS_limit 100
#define DMS_max 3
#define DMS 3
#define cpwidth 2/*3*/
#define CPMAX /*8*//*16*//*27*//*54*/16
#define _3d14 /*-2*//*-1*//*0*//*1*/1
#define Ks (1+3)
#define K_ 3                        /* >0:auto, 0:manual */

#define VGACOLORS 64
#define fcolor -1
#define wall -2
#define delta_x 2
#define RESO (cpwidth*(2*delta_x+1))
```

```
#define XRESO 1280
#define YRESO 768
#define Zx (XRESO*2)
#define Zy (YRESO*2)
#define CNT 1

#define ICEIL(a,b) (((a)+((b)-1))/(b))
#define UdX 10
#define UdY 20
#define TORAD (acos(-1)/180)

#define ASIZE (256+1)

char Rectflag;
int idx,idx_Rect;
char work_Rect[Zx][Zy];

int Zflag,dO[2];
long rho,dscr;
dbl th,ph,mx[4][3];
dbl DET,xE,yE,zE;
dbl Zbuf[Zx][Zy];

char refill,pauseflag,fieldflag,GRPH,c_trans,plflag,pl[CPMAX];
char charcode,charflag,Trianflag,SPmode;
/* here */
int X,Y,Z,X_,Y_,Z_,xg,yg,zg,ui;
/* here */
int ca,c1,c2,c3,c4,c9,c10;
int c5,c6,c7,c8,c11,c12,c13,c14,c15,c16,c17,c18,c19,c20,c21,c22,c23,c24,c25,c26;
int acolor[CPMAX*1];
/* here */
int nx[Ks][CPMAX],ny[Ks][CPMAX],nz[Ks][CPMAX],nx_[Ks][CPMAX],ny_[Ks][CPMAX],nz_[Ks][CPMAX];
int plx_[CPMAX],ply_[CPMAX],plz_[CPMAX];
/* here */
int Nx,Ny,Nz,Nx_,Ny_,Nz_;
/* here */
int x[1+26],y[1+26],z[1+26],x_[1+26],y_[1+26],z_[1+26];
/* here */
int enX[8],enY[8],enZ[8],enX_[8],enY_[8],enZ_[8],in[8],cc[8];
int ig,PS=8,putperiod,sn_,CPMAXh;
/* here */
int xt,yt,zt,nax[CPMAX*1],nay[CPMAX*1],naz[CPMAX*1];
long pcount[CPMAX],cnt,sum;
dbl tmp0,tmp1,tmp2;

char pixel[Ks][RESO][RESO][RESO];

char function,usflag;
unsigned char yorn;
int WB;
long jpow[DMS_max];
FILE *fp[CPMAX];

#if WX==1
```

```
int argc_,asct;
char **argv_,appliname[]="CAG",fs1[ASIZE];

char *fs2[5]={
              "-*-*-medium-r-normal--16*",       /* 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 {
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];
/* here */
typedef struct {short xx,yy,zz,
                      xx_,yy_,zz_;
                char pl;} sss;
sss ss;
#if WX==0
typedef struct {
unsigned char red,green,blue;} srgb;
srgb irgb[VGACOLORS];
typedef struct {
unsigned long back_;int back,fore;} bf;
bf bfset[]={{WHITENESS,15,0},{BLACKNESS,0,15}};
#else
XColor irgb[VGACOLORS],c;
typedef struct {
int back,fore;} bf;
bf bfset[]={{15,0},{0,15}};
#endif

#if WX==0
HINSTANCE hinstance;
HWND hwnd;
HDC hdcdisplay,hdctmp1;
HBITMAP hbitmap1;
#else
Display *d;
int screen,depth;
Colormap cmap;
Window rw,ww;
XSizeHints sh;
GC gcdisplay;
Pixmap pmap1;
XFontSet font_fs;
XFontStruct **info;
Cursor cursor;
XEvent event;
```

```
KeySym keysym,sym;
Atom atm1,atm2;
Visual *vis;
XImage *image;

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

void closegraph_(void),initpalette(void),BitBlt_full(void),terminator(int),
     cleardevice_(char,int,int,int,int),rectangle_(char,int,int,int,int,int,int),
     delay_(long),beep(long),kbhit_(void),restore_3(void),check_rcount(void),
     use_subroop(void),keydowns_f2(void),bitblt(char,int,int,int,int,int,int),
     arrayreset(void),field(void),
     axes_eye(char,char,char,dbl,dbl,dbl,char*,char*,char*),
     line_eye_(dbl,dbl,dbl,dbl,dbl,dbl,int),mallocs(void),frees(void),getsum(void),
     initeye(void),set_O(int,int),clearZbuf(void),set_Z(char),
     projection(dbl,dbl,dbl,int *,int *),puts_(int,int,int,char *);
unsigned char subroop(void);
int initgraph_(void),fourfloor_fiveceil(dbl),random_(int),cag_r(long),
    putpixel(int,int,int,int);
    getpixel(int,int,int),
    rpixel(int,int,int);

#if WX==0
COLORREF PALETTE(int color);
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)
{
char str[32];
int i,xs,ys,dlt,xo,yo,val;
long mytime,oldtime,nowtime;
dbl x,y,z;

if(DMS<3 || DMS>DMS_max) return 1;



fieldflag=-1;

if(!Xlib){
GRPH=0;
```

```
if(argc>1 && strcmp(argv[1],"0")==0) {if(argc==2) argc=1;else argc=2;}
else     {if(argc==1) argc=1;else argc=2;}  /* dr (s) */
}
else if(argc>1 && strcmp(argv[1],"0")==0){
GRPH=0;
if(argc==2) argc=1;else argc=2;
}
else{
GRPH=1;
}
WB=0;
refill=1;
sn_=0;

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

#if !WX || Xlib
if(GRPH){
rho=5000*1;th=60;ph=-40.5;
dscr=5000;

initeye();
set_Z(1);
if(DMS==3) {xo=80+50;yo=205+185-80;}
/*if(DMS==3) {xo=80+25;yo=205+185-140;}*/
else       {xo=80;yo=205;}
set_O(0+xo,0+yo);

     if(DMS==3) sn_=-1;
else if(DMS>3 && Xlib==2 && GRPH/* && cnt==CNT-1*/) sn_=1;
}

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

printf(" DMS:%dd/%dd PBC:%d CPMAX:%d delta_x:%d RESO:%d\n",DMS,DMS_max,PBC,CPMAX,delta_x,RESO);

if(argc>1) {time(&mytime);srand((unsigned int)mytime);}
else
srand(/*1*/6-1);

xt=RESO-1;
yt=RESO-1;
zt=RESO-1;
CPMAXh=pow(cpwidth,3);

mallocs();
for(i=1;i<DMS_max;i++) jpow[i]=pow(RESO,i);
arrayreset();

printf(" \n");

while(1){
time(&nowtime);
```

```
strcpy(str,ctime(&nowtime));
i=0;
while(1){
if(str[i]==':') break;
i++;
}
strncpy(str,&str[i-2],5);
str[5]='\0';
printf(" %s\n",str);

if(DMS>3 && Xlib==2 && GRPH/* && cnt==CNT-1*/){
cleardevice_(1,0,0,XRESO,YRESO);

clearZbuf();
axes_eye(1+0,1+0,1+0,110,170,100,"x","y","z");
projection(0,0,0,&xs,&ys);
puts_(16,xs-1.2*UdX,ys-0.3*UdY,"O");

val=4;

clearZbuf();
set_O(0+xo,0+yo);
projection((x=0),(y=val*RESO*PS),(z=0),&xs,&ys);
set_O(0+xs,0+ys);
axes_eye(1+0,1+0,1+0,110,170,100,"x","y","z");
puts_(16,xs-1.2*UdX,ys-0.3*UdY,"O");

clearZbuf();
set_O(0+xo,0+yo);
projection((x=val*RESO*PS),(y=0),(z=0),&xs,&ys);
set_O(0+xs,0+ys);
axes_eye(1+0,1+0,1+0,110,170,100,"x","y","z");
puts_(16,xs-1.2*UdX,ys-0.3*UdY,"O");

clearZbuf();
set_O(0+xo,0+yo);
projection((x=val*RESO*PS),(y=val*RESO*PS),(z=0),&xs,&ys);
set_O(0+xs,0+ys);
axes_eye(1+0,1+0,1+0,110,170,100,"x","y","z");
puts_(16,xs-1.2*UdX,ys-0.3*UdY,"O");

clearZbuf();
set_O(0+xo,0+yo);

BitBlt_full();
}
else if(DMS==3 && Xlib && GRPH){
cleardevice_(1,0,0,XRESO,YRESO);
clearZbuf();
axes_eye(1+0,1+0,1+0,110+135,170+150,100+135,"X","Y","Z");
BitBlt_full();
}

time(&oldtime);
```

```
if(refill==0) break;
field();
if(refill==0) break;
#if !PBC
if(cnt==0) getsum();
#endif
printf(" sum:%ld _3d14:%d K_:%d\n",sum,_3d14,K_);
cag_r(oldtime);
check_rcount();

printf(" \n");
if(refill<=0) break;

if(GRPH){
beep(50);

delay_(6000);
if(pauseflag==1) {pauseflag=0;use_subroop();}
}/**if(GRPH)**/
}/**while(1)**/

closegraph_();

return 0;
}/** main **/


void fprintf_(char flag,int i,int plane,int algo,char *str)
{
FILE *fp_;

/*if(!flag || cnt<CNT || pcount[CPMAX-1]<RCNT-1) return;*/
if(cnt!=CNT) return;

if(flag==1){
#if 0
if(algo==-1) fprintf(fp_," cnt:%d %d %d\n",i,-1,-1);
#else
fp_=fopen("Cpage.bin","ab");
if(i==0)
fprintf(fp_," %ld %ld i:%d plane:%d algo:%d%s\n",cnt,pcount[0]+1,i,plane,algo%2,str);
else if(i==CPMAX-1)
fprintf(fp_," %ld %ld i:%d plane:%d algo:%d%s\n\n",cnt,pcount[0],i,plane,algo%2,str);
else
fprintf(fp_," %ld %ld i:%d plane:%d algo:%d%s\n",cnt,pcount[0],i,plane,algo%2,str);
fclose(fp_);
#endif
}
else if(flag==2){
if(i==0)
printf(" %ld %ld i:%d plane:%d algo:%d%s\n",cnt,pcount[0]+1,i,plane,algo%2,str);
else if(i==CPMAX-1)
printf(" %ld %ld i:%d plane:%d algo:%d%s\n\n",cnt,pcount[0],i,plane,algo%2,str);
else
printf(" %ld %ld i:%d plane:%d algo:%d%s\n",cnt,pcount[0],i,plane,algo%2,str);
```

```
}
else if(flag==3){
fp_=fopen("Cpage.bin","ab");
fprintf(fp_," %d %d %d %s\n",i,plane,algo,str);
fclose(fp_);
}
}/** fprintf_ **/


#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 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);*/

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){
ls_image(2,"ss.bmp",0,0,XRESO,YRESO);
printf(" S\n");
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);
printf(" S\n");
beep(300);
}
}/** keydowns_f2 **/
#endif


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

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

```
#if WX==1
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


void mallocs(void)
{
}/** mallocs **/


void frees(void)
{
}/** frees **/


int initgraph_(void)
{
```

```
#if WX==0
WNDCLASS wndclass;

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,
                  300,0,/*XRESO*/1024,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);
hdctmp1=CreateCompatibleDC(hdcdisplay);  /* text, dialog, menu */
SelectObject(hdctmp1,hbitmap1);
SetBkMode(hdcdisplay,TRANSPARENT);
SetBkMode(hdctmp1,TRANSPARENT);
SetBkColor(hdcdisplay,PALETTE(bfset[WB].back));
SetBkColor(hdctmp1,PALETTE(bfset[WB].back));
#else
if((d=XOpenDisplay(""))==NULL) return 1;
screen=DefaultScreen(d);
cmap=DefaultColormap(d,screen);

rw=DefaultRootWindow(d);
ww=XCreateSimpleWindow(d,rw,300,0,/*XRESO*/1024,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,ww,appliname,appliname,None,argv_,argc_,&sh);

atm1=XInternAtom(d,"WM_PROTOCOLS",False);
```

```
atm2=XInternAtom(d,"WM_DELETE_WINDOW",False);
XSetWMProtocols(d,ww,&atm2,1);

XSelectInput(d,ww,KeyPressMask | ButtonPressMask | PointerMotionMask | ExposureMask |
                  StructureNotifyMask | SubstructureNotifyMask);
if(GRPH) {XMapWindow(d,ww);XFlush(d);}

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

depth=DefaultDepth(d,screen);
pmap1=XCreatePixmap(d,ww,XRESO,YRESO,depth);  /* text, dialog, menu */

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

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

initsysfont(0);
#endif

initpalette();
signal(SIGINT,terminator);

return 0;
}/** initgraph_ **/


void terminator(int sig)
{
signal(sig,SIG_IGN);

printf(" Ctrl+C \n");
if(fieldflag) exit(0);
else          refill=0;
}/** terminator **/


void closegraph_(void)
{
int i;
char str[32],buf[10];

for(i=0;i<CPMAX;i++){
strcpy(str,"rtn[");
#if !WX
strcat(str,itoa(i,buf,10));
#else
strcat(str,gcvt(i,8,buf));
#endif
strcat(str,"].bin");
unlink(str);
}

frees();
```

```
#if WX==0
DeleteObject(hbitmap1);
DeleteDC(hdctmp1);

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

XFreeColormap(d,cmap);
XDestroyWindow(d,ww);XFlush(d);
XCloseDisplay(d);
#endif
}/** closegraph_ **/


void initpalette(void)
{
#if WX==0 || Xlib
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=irgb[9+(i-1)].red-24*1;
if(irgb[9+(i-1)].green==255)
irgb[i].green=irgb[9+(i-1)].green-24*1;
if(irgb[9+(i-1)].blue==255)
irgb[i].blue=irgb[9+(i-1)].blue-24*1;
}

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

if(DMS==3 && CPMAX>64){

for(i=16;i<CPMAX;i++){
```

```
irgb[i].red=255*((dbl)CPMAX-(i-16))/CPMAX;
irgb[i].green=irgb[i].red;
irgb[i].blue=irgb[i].red;
}

}/**if(DMS,CPMAX)**/
else{

if(CPMAX<=64){
for(i=16;i<64;i++){
irgb[i].red=255*(64.-(i-16))/64;
irgb[i].green=irgb[i].red;
irgb[i].blue=irgb[i].red;
}
}
else{
for(i=16;i<VGACOLORS;i++){
irgb[i].red=255*((dbl)VGACOLORS-(i-16))/VGACOLORS;
irgb[i].green=irgb[i].red;
irgb[i].blue=irgb[i].red;
}
}

}/**else(DMS,CPMAX)**/
#endif

#if WX==1
#if Xlib
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
#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_)
{
BitBlt(hdcdisplay,x_,y_,xsize,ysize,
       hdctmp1,x,y,SRCCOPY);
```

```
}/** bitblt **/
#else
void bitblt(char flag,int x,int y,int xsize,int ysize,int x_,int y_)
{
XCopyArea(d,pmap1,ww,gcdisplay,x,y,xsize,ysize,
                               x_,y_);

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


#if WX==0
void cleardevice_(char flag,int x,int y,int xsize,int ysize)
{
PatBlt(hdctmp1,x,y,xsize,ysize,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);

XFillRectangle(d,pmap1,gcdisplay,x,y,xsize,ysize);
}/** cleardevice_ **/
#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)
{
int i;

if(XPending(d)){
XNextEvent(d,&event);

if(event.type==ClientMessage &&
   event.xclient.message_type==atm1 && event.xclient.data.l[0]==atm2){
printf(" Close\n");

if(fieldflag) exit(0);
else          refill=0;
}
else{
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)
{
if(umsg==WM_KEYDOWN){
/************************** menu keydowns -> ****************************/
/************************** <- menu keydowns ****************************/

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

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

if(usflag==1) usflag=0;

     if(GKS(VK_ESCAPE)<0 || GKS(VK_PAUSE)<0) refill=0;
```

```
else if(GKS_(VK_SHIFT)<0) pauseflag=1;

return 1;
}/**else if(umsg)**/
else if(umsg==WM_SYSKEYDOWN){
}/**else if(umsg)**/
else if(umsg==WM_CLOSE){
if(fieldflag) exit(0);
else          refill=0;

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

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

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

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

return 0;
}/** wndproc_filer **/
#else
int wndproc_filer(void)
{
int length;
/*static */char buf[10];

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

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

/************************** dialog keydowns -> ****************************/
/************************** <- 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){

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

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


void delay_(long millisecond)
{
long oldtime,nowtime,dtime=0,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 **/


dbl det(dbl a11,dbl a12,dbl a13,dbl a21,dbl a22,dbl a23,
           dbl a31,dbl a32,dbl a33)
{
dbl val;

val=a11*a22*a33+a12*a23*a31+a13*a32*a21-a13*a22*a31-a12*a21*a33-a11*a32*a23;

return val;
}/** det **/


void initeye(void)
{
dbl sinth,costh,sinph,cosph;

if(th==0) th=360;

th*=TORAD;ph*=TORAD;
sinth=sin(th);costh=cos(th);
sinph=sin(ph);cosph=cos(ph);

mx[0][0]=-sinph;
mx[1][0]=cosph;
mx[2][0]=0;

mx[0][1]=-costh*cosph;
mx[1][1]=-costh*sinph;
mx[2][1]=sinth;

mx[0][2]=-sinth*cosph;
mx[1][2]=-sinth*sinph;
mx[2][2]=-costh;

mx[3][0]=0;
mx[3][1]=0;
mx[3][2]=rho;

xE=rho*sinth*cosph;
yE=rho*sinth*sinph;
zE=rho*costh;

DET=det(mx[0][0],mx[1][0],mx[2][0],mx[0][1],mx[1][1],mx[2][1],mx[0][2],mx[1][2],
        mx[2][2]);
}/** initeye **/
```

```
void set_O(int x,int y)
{
dO[0]=x;dO[1]=y;
}/** set_O **/


void projection(dbl x,dbl y,dbl z,int *xs,int *ys)
{
/*int dx,dy;*/
dbl xe,ye,ze;

/*dx=XRESO/2;dy=YRESO/2;*/

xe=mx[0][0]*x+mx[1][0]*y;
ye=mx[0][1]*x+mx[1][1]*y+mx[2][1]*z;
ze=mx[0][2]*x+mx[1][2]*y+mx[2][2]*z+mx[3][2];

*xs=ff_fc(dscr*xe/ze)+/*dx*/dO[0];
*ys=ff_fc(-dscr*ye/ze)+/*dy*/dO[1];
}/** projection **/


void clearZbuf(void)
{
int i,j;

for(j=0;j<Zy;j++)
for(i=0;i<Zx;i++){
Zbuf[i][j]=rho*5;
work_Rect[i][j]=0;
work[i][j].p=0;
}

idx=0;
idx_Rect=0;
}/** clearZbuf **/


void set_Z(char flag)
{
Zflag=flag;
}/** set_Z **/


#if WX==0
void puts_(int h,int x,int y,char *str)
{
HFONT hfont;

hfont=CreateFont(h,h/2,0,0,
                 FW_NORMAL,/*TRUE*/0,0,0,
                 DEFAULT_CHARSET,OUT_DEFAULT_PRECIS,
                 CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,
```

```
                FIXED_PITCH | FF_ROMAN/*FF_MODERN*/,NULL);

SetTextColor(hdctmp1,RGB(0,0,0));
TextOut(hdctmp1,x,y,str,strlen(str));

DeleteObject(hfont);
}/** puts_ **/
#else
void puts_(int h,int x,int y,char *str)
{
XSetForeground(d,gcdisplay,irgb[0].pixel);

XmbDrawString(d,pmap1,font_fs,gcdisplay,x,y+XDSDY,str,strlen(str));
}/** puts_ **/
#endif


void axes_eye(char flagx,char flagy,char flagz,dbl lenx,dbl leny,dbl lenz,
              char *xname,char *yname,char *zname)
{
int xs,ys,dx,dy,ds,ds0=15,dys=8;

if(flagx>0) line_eye_(0,0,0,lenx,0,0,0);
if(flagy>0) line_eye_(0,0,0,0,leny,0,0);
if(flagz>0) line_eye_(0,0,0,0,0,lenz,0);

projection(lenx,0,0,&xs,&ys);
dx=xs-dO[0];
dy=ys-dO[1];
ds=sqrt(dx*dx+dy*dy);
if(ds)
puts_(16,xs+dx*ds0/ds-dys/2,ys+dy*ds0/ds-dys,xname);
else
puts_(16,xs+ds0,ys-dys,xname);

projection(0,leny,0,&xs,&ys);
dx=xs-dO[0];
dy=ys-dO[1];
ds=sqrt(dx*dx+dy*dy);
if(ds)
puts_(16,xs+dx*ds0/ds-dys/2,ys+dy*ds0/ds-dys,yname);
else
puts_(16,xs+ds0,ys-dys,yname);

projection(0,0,lenz,&xs,&ys);
dx=xs-dO[0];
dy=ys-dO[1];
ds=sqrt(dx*dx+dy*dy);
if(ds)
puts_(16,xs+dx*ds0/ds-dys/2,ys+dy*ds0/ds-dys,zname);
else
puts_(16,xs+ds0,ys-dys,zname);
}/** axes_eye **/
```

```
void check_rcount(void)
{
int i,j;
long val[2],vl[4],sum;

if(GRPH>0){
for(j=0;j<CPMAX;j++)
printf(" %ld %ld\n",cnt,pcount[j]);
}
else{
#if /*CPMAX<=CPMAXh*2 && _3d14>=0*/0
val[0]=pcount[0];
for(j=1;j<CPMAX;j++){
if(pcount[j]!=val[0]) {beep(1000);refill=-1;break;}
}
#elif 0
if(CPMAX==CPMAXh*2){
val[0]=pcount[0];
for(j=1;j<CPMAXh;j++){
if(pcount[j]!=val[0]) {beep(1000);refill=-1;break;}
}
val[1]=pcount[CPMAXh];
for(j=CPMAXh;j<CPMAX;j++){
if(pcount[j]!=val[1]) {beep(1000);refill=-1;break;}
}

}
else{
val[0]=pcount[0];
for(j=1;j<CPMAX;j++){
if(pcount[j]!=val[0]) {beep(1000);refill=-1;break;}
}
}
#else
_1st:
val[0]=pcount[0];pcount[0]*=-1;
i=0;
for(j=1;j<CPMAX;j++){
if(pcount[j]==val[0]){
i++;pcount[j]*=-1;
if(i==CPMAXh-1) {vl[0]=val[0];if(CPMAX==CPMAXh*1) goto next;else goto _2nd;}
}
}
refill=-1;goto next;

_2nd:
for(j=1;j<CPMAX;j++){
if(pcount[j]>0) {val[0]=pcount[j];pcount[j]*=-1;break;}
}
i=0;
for(j=1;j<CPMAX;j++){
if(pcount[j]==val[0]){
i++;pcount[j]*=-1;
if(i==CPMAXh-1) {vl[1]=val[0];if(CPMAX==CPMAXh*2) goto next;else goto _3rd;}
}
```

```
}
refill=-1;goto next;

_3rd:
for(j=1;j<CPMAX;j++){
if(pcount[j]>0) {val[0]=pcount[j];pcount[j]*=-1;break;}
}
i=0;
for(j=1;j<CPMAX;j++){
if(pcount[j]==val[0]){
i++;pcount[j]*=-1;
if(i==CPMAXh-1) {vl[2]=val[0];goto _4th;}
}
}
refill=-1;goto next;

_4th:
for(j=1;j<CPMAX;j++){
if(pcount[j]>0) {val[0]=pcount[j];pcount[j]*=-1;break;}
}
i=0;
for(j=1;j<CPMAX;j++){
if(pcount[j]==val[0]){
i++;pcount[j]*=-1;
if(i==CPMAXh-1) {vl[3]=val[0];goto next;}
}
}
refill=-1;

next:
for(j=0;j<CPMAX;j++) {if(pcount[j]<0) pcount[j]*=-1;}
     if(CPMAX==CPMAXh*1) {vl[1]=vl[2]=vl[3]=0;}
else if(CPMAX==CPMAXh*2) {vl[2]=vl[3]=0;}
#endif

if(refill==0){
}
else if(refill==-1){
printf(" %dd %ld %ld %d:%ld\n",DMS,cnt,val[0],j,pcount[j]);
}
else if(refill<=-2){
printf(" %dd %d %ld %ld\n",DMS,refill,cnt,val[0]);
}
else{
#if PBC
#if /*CPMAX<=CPMAXh*2 && _3d14>=0*/0
if(0) for(j=0;j<CPMAX;j++) printf(" %ld %ld\n",cnt,pcount[j]);
else  printf(" %ddi %dCP ok:%ld %ld\n",DMS,CPMAX,cnt,val[0]);
#else
if(CPMAX>=CPMAXh*2)
printf(" %ddi %dCP ok:%ld %ld %ld %ld %ld\n",DMS,CPMAX,cnt,vl[0],vl[1],vl[2],vl[3]);
else
printf(" %ddi %dCP ok:%ld %ld\n",DMS,CPMAX,cnt,val[0]);
#endif
#else
```

```
printf(" %ddf %dCP ok:%ld %ld\n",DMS,CPMAX,cnt,val[0]);
#endif
}
}

end:
/*if(!GRPH && cnt==GRPH_0_MAX) {beep(100);refill=0;}*/
if(refill<0){
for(j=0;j<CPMAX;j++) printf(" %ld\n",pcount[j]);
}

sum=0;
for(j=0;j<CPMAX;j++) sum+=pcount[j];
printf(" %ld\n",sum);
}/** check_rcount **/


void arrayreset(void)
{
int i,j,k/*,l*/;
int _t[DMS_limit];

_t[0]=xt;
_t[1]=yt;
_t[2]=zt;
/*_t[3]=ut;*/

k=0;
while(1){

j=0;
while(1){

/* here */
i=0;
while(1){
/* here */
pixel[0][i][j][k]=wall;
pixel[1][i][j][k]=wall;
pixel[2][i][j][k]=wall;
pixel[3][i][j][k]=wall;

/* here */
i++;
if(i==_t[0]+1) break;
}

j++;
if(j==_t[1]+1) break;
}

k++;
if(k==_t[2]+1) break;
}
}/** arrayreset **/
```

```
#if WX==0
int ppixel(int nx,int ny,int pcolor)
{
int grp;

if(sn_){
if((nx<0)||(nx>XRESO-1)||(ny<0)||(ny>YRESO-1)) return 1;
/*if((nx<0)||(nx>XRESO-1)||(ny<0)||(ny>YrESO-1)) return 1;*/

if(Trianflag){
grp=ig/CPMAXh;
if(grp==0) pcolor=15;
else       pcolor=9+grp-2;
}

SetPixelV(hdcdisplay,nx,ny,PALETTE(pcolor));
SetPixelV(hdctmp1,nx,ny,PALETTE(pcolor));
}

return 0;
}/** ppixel **/
#else
int ppixel(int nx,int ny,int pcolor)
{
#if Xlib
int grp;

if(sn_){
if((nx<0)||(nx>XRESO-1)||(ny<0)||(ny>YRESO-1)) return 1;
/*if((nx<0)||(nx>XRESO-1)||(ny<0)||(ny>YrESO-1)) return 1;*/

if(Trianflag){
grp=ig/CPMAXh;
if(grp==0) pcolor=15;
else       pcolor=9+grp-2;
}
XSetForeground(d,gcdisplay,irgb[pcolor].pixel);

XDrawPoint(d,ww,gcdisplay,nx,ny);
XDrawPoint(d,pmap1,gcdisplay,nx,ny);
}
#endif

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


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 **/


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) ppixel(xs,ys,color);
else{
/*xb=x-dO[0]+Zx/2;
yb=y-dO[1]+Zy/2;*/
if(i<0 || i>Zx-1 || y<0 || y>Zy-1) ;
else{
val=1.*(i-left)/(right-left);
X=work[left][y].x+DX*val;
Y=work[left][y].y+DY*val;
Z=work[left][y].z+Dz*val;
len=sqrt(pow(xE-X,2)+pow(yE-Y,2)+pow(zE-Z,2));

if(len<Zbuf[i][y] || Zflag==2){
if(Zflag==1) Zbuf[i][y]=len;
if(work_Rect[i][y]==0) ppixel(xs,ys,color);
}

}
}
}/**for(i)**/
}/** hline **/
```

```
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,ds,tmp0,tmp1,tmp2;

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

xg=x2;yg=y2;

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);
ds=sqrt(dx*dx+dy*dy);

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{
val=0;
X=xd1+DX*val;
Y=yd1+DY*val;
Z=zd1+Dz*val;
len=sqrt(pow(xE-X,2)+pow(yE-Y,2)+pow(zE-Z,2));

if(len<Zbuf[xb][yb] || Zflag==2){
if(Zflag==1) Zbuf[xb][yb]=len;
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-dO[0]+Zx/2;
yb=y-dO[1]+Zy/2;
if(xb<0 || xb>Zx-1 || yb<0 || yb>Zy-1) ;
else{
val=sqrt(pow(x-x1,2)+pow(y-y1,2))/ds;
X=xd1+DX*val;
Y=yd1+DY*val;
Z=zd1+Dz*val;
len=sqrt(pow(xE-X,2)+pow(yE-Y,2)+pow(zE-Z,2));
```

```
if(len<Zbuf[xb][yb] || Zflag==2){
if(Zflag==1) Zbuf[xb][yb]=len;
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{
val=sqrt(pow(x-x1,2)+pow(y-y1,2))/ds;
X=xd1+DX*val;
Y=yd1+DY*val;
Z=zd1+Dz*val;
len=sqrt(pow(xE-X,2)+pow(yE-Y,2)+pow(zE-Z,2));

if(len<Zbuf[xb][yb] || Zflag==2){
if(Zflag==1) Zbuf[xb][yb]=len;
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 line_thph_eye(dbl th,dbl ph,dbl len,dbl x1,dbl y1,dbl z1,
                   int color)
{
char flag=1;
dbl x2,y2,z2;

if(len==0) return;
if(len<0) {flag=0;len=-len;}

x2=x1+len*sin(th*TORAD)*cos(ph*TORAD);
y2=y1+len*sin(th*TORAD)*sin(ph*TORAD);
z2=z1+len*cos(th*TORAD);
```

```
if(color>=0) line_eye_(x1,y1,z1,x2,y2,z2,color);

if(flag==1){
/*xg=x2;yg=y2;zg=z2;*/
tmp0=x2;tmp1=y2;tmp2=z2;
}
}/** line_thph_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){
if(color3<0) {Trianflag=1;color3=0;}else Trianflag=0;
line_eye_(x1,y1,z1,x2,y2,z2,color3);
if(color1<0) {Trianflag=1;color1=0;}else Trianflag=0;
line_eye_(x2,y2,z2,x3,y3,z3,color1);
if(color2<0) {Trianflag=1;color2=0;}else Trianflag=0;
line_eye_(x3,y3,z3,x1,y1,z1,color2);
Trianflag=0;
}
else{
line_eye_(x1,y1,z1,x2,y2,z2,color3);
line_eye_(x2,y2,z2,x3,y3,z3,color1);
line_eye_(x3,y3,z3,x1,y1,z1,color2);
return;
}

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){
if(/*ig!=15*/1) hline(xmin_,xmax_,y,color);
}
}/**for(y)**/
}/** Trian **/


void Rect(dbl x1,dbl y1,dbl z1,dbl x2,dbl y2,dbl z2,
          dbl x3,dbl y3,dbl z3,dbl x4,dbl y4,dbl z4,int color)
{
restore_work(1);

Rectflag=1;
if((ig/CPMAXh)!=1){
Trian(0,x3,y3,z3,x2,y2,z2,x1,y1,z1,-1,color,-1,color);
Trian(0,x1,y1,z1,x3,y3,z3,x4,y4,z4,-1,-1,color,color);
}
else{
Trian(0,x3,y3,z3,x2,y2,z2,x1,y1,z1,0,color,0,color);
Trian(0,x1,y1,z1,x3,y3,z3,x4,y4,z4,0,0,color,color);
}
Rectflag=0;
}/** Rect **/


void polygon(dbl vx[],dbl vy[],dbl vz[],int color)
{
Rect(vx[0],vy[0],vz[0],vx[1],vy[1],vz[1],vx[2],vy[2],vz[2],vx[3],vy[3],vz[3],color);
}/** polygon **/
```

```
void pixel_3d(int nx,int ny,int nz,int color)
{
dbl vx[4],vy[4],vz[4];

if(!GRPH) return;

nx*=2;ny*=2;nz*=2;
if(pl[ig]){
     if(pl[ig]==1) ny+=PS*45;
else if(pl[ig]==2) nx+=PS*45;
else               {nx+=PS*45;ny+=PS*45;}
}

if(1){
/* -z */
vx[0]=nx;vy[0]=ny;vz[0]=nz;
vx[1]=nx+PS;vy[1]=ny;vz[1]=nz;
vx[2]=nx+PS;vy[2]=ny+PS;vz[2]=nz;
vx[3]=nx;vy[3]=ny+PS;vz[3]=nz;
polygon(vx,vy,vz,color);

/* +z */
vx[0]=nx;vy[0]=ny;vz[0]=nz+PS;
vx[1]=nx+PS;vy[1]=ny;vz[1]=nz+PS;
vx[2]=nx+PS;vy[2]=ny+PS;vz[2]=nz+PS;
vx[3]=nx;vy[3]=ny+PS;vz[3]=nz+PS;
polygon(vx,vy,vz,color);

/* -y */
vx[0]=nx;vy[0]=ny;vz[0]=nz;
vx[1]=nx;vy[1]=ny;vz[1]=nz+PS;
vx[2]=nx+PS;vy[2]=ny;vz[2]=nz+PS;
vx[3]=nx+PS;vy[3]=ny;vz[3]=nz;
polygon(vx,vy,vz,color);

/* +y */
vx[0]=nx;vy[0]=ny+PS;vz[0]=nz;
vx[1]=nx;vy[1]=ny+PS;vz[1]=nz+PS;
vx[2]=nx+PS;vy[2]=ny+PS;vz[2]=nz+PS;
vx[3]=nx+PS;vy[3]=ny+PS;vz[3]=nz;
polygon(vx,vy,vz,color);
}

if(1){
/* -x */
vx[0]=nx;vy[0]=ny;vz[0]=nz;
vx[1]=nx;vy[1]=ny+PS;vz[1]=nz;
vx[2]=nx;vy[2]=ny+PS;vz[2]=nz+PS;
vx[3]=nx;vy[3]=ny;vz[3]=nz+PS;
polygon(vx,vy,vz,color);

/* +x */
vx[0]=nx+PS;vy[0]=ny;vz[0]=nz;
vx[1]=nx+PS;vy[1]=ny+PS;vz[1]=nz;
```

```
vx[2]=nx+PS;vy[2]=ny+PS;vz[2]=nz+PS;
vx[3]=nx+PS;vy[3]=ny;vz[3]=nz+PS;

polygon(vx,vy,vz,color);
}

/*bitblt(1,min_x,min_y,max_x-min_x+1,max_y-min_y+1,min_x,min_y);*/
}/** pixel_3d **/


#if !PBC
void getsum(void)
{
/* here * */  /* add p */
int x,y,z;

/* here * *//* add*/
for(z=0;z<=zt;z++)
for(y=0;y<=yt;y++)
for(x=0;x<=xt;x++){
/* here * */         /* add p, */
if(pixel[x][y][z]==fcolor) sum++;
}
}/** getsum **/
#endif


void field(void)
{
int x,y,z,/*fcolor=16,*/old;

fieldflag=1;
old=sn_;sn_=0;

/* here **/
if(0){
}/**if()**/
else{
/* here */
/* all */
for(z=0;z<=zt;z++)
for(y=0;y<=yt;y++)
for(x=0;x<=xt;x++){
putpixel(x,y,z,fcolor);
if(cnt==0) sum++;
}
}

end:

sn_=old;
fieldflag=0;
}/** field **/
```

```
int putpixel(int cx,int cy,int cz,int pcolor)
{
int i,j,k,plane;

/* here */
/*if((cx<0)||(cx>RESO-1)||(cy<0)||(cy>RESO-1)||(cz<0)||(cz>RESO-1)) return 1;*/
if(!GRPH) goto end;

     if(/*sn_==0*/0) {pixel_3d(cx*PS,cy*PS,cz*PS,8);/*use_subroop();*/}
else if(sn_==-1) {pixel_3d(cx*PS,cy*PS,cz*PS,pcolor);/*use_subroop();*/}
else{
}

end:
/* here */
if(fieldflag){
pixel[0][cx][cy][cz]=pcolor;
if(1) pixel[1][cx][cy][cz]=pcolor;
if(1) pixel[2][cx][cy][cz]=pcolor;
if(1) pixel[3][cx][cy][cz]=pcolor;
}
else{
pixel[pl[ig]][cx][cy][cz]=pcolor;
}

return 0;
}/** putpixel **/


void putpixel_(int cx,int cy,int cz,int pcolor)
{
putpixel(cx,cy,cz,pcolor);
}/** putpixel_ **/


int getpixel_(int nx,int ny,int nz)
{
/* here */
X=nx;Y=ny;Z=nz;
X_=Nx;Y_=Ny;Z_=Nz;

/* here */
if((nx<0)||(nx>RESO-1)||(ny<0)||(ny>RESO-1)||(nz<0)||(nz>RESO-1)) return wall;

/* here */
return pixel[pl[ig]][nx][ny][nz];
}/** getpixel_ **/


int getpixel_pl(int dx,int dy,int dz)
{
char pxl,old;
int nx,ny,nz,dn;
dbl val;
```

```
old=pl[ig];

#if /*_3d14==-1*/0
if(old>0){                          /* nx:new, Nx:old */
if(dy==0){
     if(dx==1 && dz==1) {nx=Nx+1;ny=Ny;nz=Nz+1;}
else if(dx==-1 && dz==1) {nx=Nx;ny=Ny;nz=Nz+1;}
else if(dx==1 && dz==-1) {nx=Nx+1;ny=Ny;nz=Nz;}
else if(dx==-1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz;}
}
else if(dx==0){
     if(dy==1 && dz==1) {nx=Nx;ny=Ny+1;nz=Nz+1;}
else if(dy==-1 && dz==1) {nx=Nx;ny=Ny;nz=Nz+1;}
else if(dy==1 && dz==-1) {nx=Nx;ny=Ny+1;nz=Nz;}
else if(dy==-1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz;}
}
else if(dz==0){
     if(dx==1 && dy==1) {nx=Nx+1;ny=Ny+1;nz=Nz;}
else if(dx==-1 && dy==1) {nx=Nx;ny=Ny+1;nz=Nz;}
else if(dx==1 && dy==-1) {nx=Nx+1;ny=Ny;nz=Nz;}
else if(dx==-1 && dy==-1) {nx=Nx;ny=Ny;nz=Nz;}
}
}/**if(old)**/
#else
if(old>0){                          /* nx:new, Nx:old */
if(dy==0){
if(old==2){
     if(dx==1 && dz==1) {nx=Nx;ny=Ny;nz=Nz+1;}
else if(dx==-1 && dz==1) {nx=Nx-1;ny=Ny;nz=Nz+1;}
else if(dx==1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz;}
else if(dx==-1 && dz==-1) {nx=Nx-1;ny=Ny;nz=Nz;}
}
else if(old==3){
     if(dx==1 && dz==1) {nx=Nx+1;ny=Ny;nz=Nz;}
else if(dx==-1 && dz==1) {nx=Nx;ny=Ny;nz=Nz;}
else if(dx==1 && dz==-1) {nx=Nx+1;ny=Ny;nz=Nz-1;}
else if(dx==-1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz-1;}
}
else if(old==1){                     /* 3d.14 */
     if(dx==1 && dz==1) {nx=Nx+1;ny=Ny;nz=Nz+1;}
else if(dx==-1 && dz==1) {nx=Nx;ny=Ny;nz=Nz+1;}
else if(dx==1 && dz==-1) {nx=Nx+1;ny=Ny;nz=Nz;}
else if(dx==-1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz;}
}
}
else if(dx==0){
if(old==1){
     if(dy==1 && dz==1) {nx=Nx;ny=Ny;nz=Nz+1;}
else if(dy==-1 && dz==1) {nx=Nx;ny=Ny-1;nz=Nz+1;}
else if(dy==1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz;}
else if(dy==-1 && dz==-1) {nx=Nx;ny=Ny-1;nz=Nz;}
}
else if(old==3){
     if(dy==1 && dz==1) {nx=Nx;ny=Ny+1;nz=Nz;}
else if(dy==-1 && dz==1) {nx=Nx;ny=Ny;nz=Nz;}
```

```
else if(dy==1 && dz==-1) {nx=Nx;ny=Ny+1;nz=Nz-1;}
else if(dy==-1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz-1;}
}
else if(old==2){                    /* 3d.14 */
     if(dy==1 && dz==1) {nx=Nx;ny=Ny+1;nz=Nz+1;}
else if(dy==-1 && dz==1) {nx=Nx;ny=Ny;nz=Nz+1;}
else if(dy==1 && dz==-1) {nx=Nx;ny=Ny+1;nz=Nz;}
else if(dy==-1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz;}
}
}
else if(dz==0){
if(old==1){
     if(dx==1 && dy==1) {nx=Nx+1;ny=Ny;nz=Nz;}
else if(dx==-1 && dy==1) {nx=Nx;ny=Ny;nz=Nz;}
else if(dx==1 && dy==-1) {nx=Nx+1;ny=Ny-1;nz=Nz;}
else if(dx==-1 && dy==-1) {nx=Nx;ny=Ny-1;nz=Nz;}
}
else if(old==2){
     if(dx==1 && dy==1) {nx=Nx;ny=Ny+1;nz=Nz;}
else if(dx==-1 && dy==1) {nx=Nx-1;ny=Ny+1;nz=Nz;}
else if(dx==1 && dy==-1) {nx=Nx;ny=Ny;nz=Nz;}
else if(dx==-1 && dy==-1) {nx=Nx-1;ny=Ny;nz=Nz;}
}
else if(old==3){                    /* 3d.14 */
     if(dx==1 && dy==1) {nx=Nx+1;ny=Ny+1;nz=Nz;}
else if(dx==-1 && dy==1) {nx=Nx;ny=Ny+1;nz=Nz;}
else if(dx==1 && dy==-1) {nx=Nx+1;ny=Ny;nz=Nz;}
else if(dx==-1 && dy==-1) {nx=Nx;ny=Ny;nz=Nz;}
}
}
}/**if(old)**/
#endif
else{
if(dy==0){
     if(dx==1 && dz==1) {nx=Nx;ny=Ny;nz=Nz;}
else if(dx==-1 && dz==1) {nx=Nx-1;ny=Ny;nz=Nz;}
else if(dx==1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz-1;}
else if(dx==-1 && dz==-1) {nx=Nx-1;ny=Ny;nz=Nz-1;}
}
else if(dx==0){
     if(dy==1 && dz==1) {nx=Nx;ny=Ny;nz=Nz;}
else if(dy==-1 && dz==1) {nx=Nx;ny=Ny-1;nz=Nz;}
else if(dy==1 && dz==-1) {nx=Nx;ny=Ny;nz=Nz-1;}
else if(dy==-1 && dz==-1) {nx=Nx;ny=Ny-1;nz=Nz-1;}
}
else if(dz==0){
     if(dx==1 && dy==1) {nx=Nx;ny=Ny;nz=Nz;}
else if(dx==-1 && dy==1) {nx=Nx-1;ny=Ny;nz=Nz;}
else if(dx==1 && dy==-1) {nx=Nx;ny=Ny-1;nz=Nz;}
else if(dx==-1 && dy==-1) {nx=Nx-1;ny=Ny-1;nz=Nz;}
}
}/**else(old)**/

#if !PBC
if((nx<0)||(nx>RESO-1)||(ny<0)||(ny>RESO-1)||(nz<0)||(nz>RESO-1)) return wall;
```

```
X=nx;Y=ny;Z=nz;
#else
X=nx;Y=ny;Z=nz;

/* X,Y,Z:in use;X_,Y_,Z_:not in use */
if(dy==0){
     if(X==-1)   X=RESO-1;
else if(X==RESO) X=0;
/*     if(Y==-1)   Y=RESO-1;
else if(Y==RESO) Y=0;     */
     if(Z==-1)   Z=RESO-1;
else if(Z==RESO) Z=0;
}
else if(dx==0){
/*     if(X==-1)   X=RESO-1;
else if(X==RESO) X=0;     */
     if(Y==-1)   Y=RESO-1;
else if(Y==RESO) Y=0;
     if(Z==-1)   Z=RESO-1;
else if(Z==RESO) Z=0;
}
else if(dz==0){
     if(X==-1)   X=RESO-1;
else if(X==RESO) X=0;
     if(Y==-1)   Y=RESO-1;
else if(Y==RESO) Y=0;
/*     if(Z==-1)   Z=RESO-1;
else if(Z==RESO) Z=0;     */
}
#endif

if(dy==0){
if(pl[ig]==0) pl[ig]=1;
else if(pl[ig]==1) pl[ig]=0;
else if(pl[ig]==2) pl[ig]=3;
else if(pl[ig]==3) pl[ig]=2;
}
else if(dx==0){
if(pl[ig]==0) pl[ig]=2;
else if(pl[ig]==1) pl[ig]=3;
else if(pl[ig]==2) pl[ig]=0;
else if(pl[ig]==3) pl[ig]=1;
}
else if(dz==0){
if(pl[ig]==0) pl[ig]=3;
else if(pl[ig]==1) pl[ig]=2;
else if(pl[ig]==2) pl[ig]=1;
else if(pl[ig]==3) pl[ig]=0;
}

if(1) pxl=pixel[pl[ig]][X][Y][Z];
else pxl=-2;
pl[ig]=old;
```

```
return pxl;
}/** getpixel_pl **/


int jump(int i)
{
#if PBC
/* here */
if(abs(nx[pl[i]][i]-nx_[pl[i]][i])>1){            /* n?[i]>=0 */
if(nx[pl[i]][i]>nx_[pl[i]][i]) plx_[i]+=RESO;
else             plx_[i]-=RESO;
}

else if(abs(ny[pl[i]][i]-ny_[pl[i]][i])>1){
if(ny[pl[i]][i]>ny_[pl[i]][i]) ply_[i]+=RESO;
else             ply_[i]-=RESO;
}

else if(abs(nz[pl[i]][i]-nz_[pl[i]][i])>1){
if(nz[pl[i]][i]>nz_[pl[i]][i]) plz_[i]+=RESO;
else             plz_[i]-=RESO;
}
#endif

return 0;
}/** jump **/


int connect_nxpm(int nxpm)
{
#if PBC
if(nxpm==-1)        return RESO-1;
else if(nxpm==RESO) return 0;
else                return nxpm;
#else
return nxpm;
#endif
}/** connect_nxpm **/


int connect_nypm(int nypm)
{
#if PBC
if(nypm==-1)        return RESO-1;
else if(nypm==RESO) return 0;
else                return nypm;
#else
return nypm;
#endif
}/** connect_nypm **/


/* here */
int connect_nzpm(int nzpm)
{
```

```
#if PBC
if(nzpm==-1)        return RESO-1;
else if(nzpm==RESO) return 0;
else                return nzpm;
#else
return nzpm;
#endif
}/** connect_nzpm **/


int random_(int n)
{
int val;

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

return val;
}/** random_ **/


int fen(char *str,int i,int jmax)
{
int val;

if(i==jmax+1) val=0;
else if(i==-1) val=jmax;
else val=i;

/* here */
if(strcmp(str,"X")==0) return enX[val];
else if(strcmp(str,"Y")==0) return enY[val];
else if(strcmp(str,"Z")==0) return enZ[val];

else if(strcmp(str,"X_")==0) return enX_[val];
else if(strcmp(str,"Y_")==0) return enY_[val];
else if(strcmp(str,"Z_")==0) return enZ_[val];
}/** fen **/


int jen(int i,int jmax)
{
int val;

if(i==jmax+1) val=0;
else if(i==-1) val=jmax;
else val=i;

return val;
}/** jen **/


int rpixel(int i,int j,int k)
{
/* here */
if((i<0)||(i>RESO-1)||(j<0)||(j>RESO-1)||(k<0)||(k>RESO-1)){
```

```
if(c_trans==0) return wall;
else           return wall;
}

/* here */
if(c_trans==0){
return pixel[pl[ig]][i][j][k];
}
else{
}
}/** rpixel **/


void set_colors(void)
{
int i;

for(i=0;i<CPMAX;i++){
/*if(i<16) */acolor[i]=i;
/*else     acolor[i]=i+1;*/
}
}/** set_colors **/


void set_seedpoints(void)
{
int i/*,j*/,mx,my,mz/*,mu*/;
int xyz/*,bases,base_u*/;
int iend=DMS_limit-3;
int num[DMS_limit],coor[DMS_limit],db[DMS_limit];

i=0;
while(1){
db[i]=0;

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

xyz=pow(cpwidth,3);
db[0]=xyz;                          /* 2*2*2 */
i=0;
while(1){
if(db[i]==CPMAXh/cpwidth) break;
if(db[i]>CPMAXh/cpwidth) {db[i]=0;break;}

i++;if(i==iend) break;
db[i]=db[i-1]*cpwidth;
}

i=0;
while(1){
if(db[i]>0) {num[i]=cpwidth-1;coor[i]=1;}
else        {num[i]=0;coor[i]=0;}

i++;if(i==iend) break;
```

```
}

/* here */
/*base_u=0;
for(j=0;j<=num[0];j++){
mu=j%cpwidth;*/

for(i=0;i<xyz;i++){
my=i%cpwidth;
mx=(i%(cpwidth*cpwidth))/cpwidth;   /* 2, 4 or 16, 4 */
mz=i/(cpwidth*cpwidth);
/* here */
nax[i]=delta_x+mx*(2*delta_x+1);
nay[i]=delta_x+my*(2*delta_x+1);
naz[i]=delta_x+mz*(2*delta_x+1);

/*bases=base_u;
nau[bases+i]=(delta_x+mu*(2*delta_x+1))*coor[0];*/
}/**for(i)**/

/*base_u+=db[0];
}*//**for(j)**/
}/** set_seedpoints **/


void set_seedpoints_(void)
{
int i,j,mx,my,mz,mu,mv,mw,mr,ms,mt,mo,lu,ds2;

lu=0;

/* here **/
if(CPMAX==2){
for(i=0;i<4;i++){
/* lower,upper */
my=i%cpwidth;
mx=(i%(cpwidth*cpwidth))/cpwidth;
if(lu==0) mz=0;                         /* lower */
else      mz=RESO-1;                    /* upper */
nax[i]=delta_x+mx*(2*delta_x+1);nay[i]=delta_x+my*(2*delta_x+1);naz[i]=mz;
}

i=0;j=0;
nax[i]=nax[j];nay[i]=nay[j];naz[i]=naz[j];
i=1;j=3;
nax[i]=nax[j];nay[i]=nay[j];naz[i]=naz[j];
}
else if(CPMAX==4){
for(i=0;i<4;i++){
/* lower */
my=i%cpwidth;
mx=(i%(cpwidth*cpwidth))/cpwidth;
if(lu==0) mz=0;
else      mz=RESO-1;
nax[i]=delta_x+mx*(2*delta_x+1);nay[i]=delta_x+my*(2*delta_x+1);naz[i]=mz;
```

```
}

for(i=0;i<4;i++){
x[i]=nax[i];y[i]=nay[i];z[i]=naz[i];
}

i=0;j=0;
nax[i]=x[j];nay[i]=y[j];naz[i]=z[j];
i=1;j=3;
nax[i]=x[j];nay[i]=y[j];naz[i]=z[j];
if(0){
/* lower, upper */
i=2;j=1;
nax[i]=x[j];nay[i]=y[j];naz[i]=RESO-1;
i=3;j=2;
nax[i]=x[j];nay[i]=y[j];naz[i]=RESO-1;
}
else{
/* lower */
i=2;j=1;
nax[i]=x[j];nay[i]=y[j];naz[i]=z[j];
i=3;j=2;
nax[i]=x[j];nay[i]=y[j];naz[i]=z[j];
}
}
else if(CPMAX==8){
ds2=0;

for(i=0;i<8/2;i++){
my=i%cpwidth;
mx=(i%(cpwidth*cpwidth))/cpwidth;
mz=0+ds2;
nax[i]=delta_x+mx*(2*delta_x+1);nay[i]=delta_x+my*(2*delta_x+1);naz[i]=mz;
}

for(i=0;i<8/2;i++){
x[i]=nax[i];y[i]=nay[i];z[i]=naz[i];
}

/*mz=0+ds2;*/

i=0;j=0;
nax[i]=x[j];nay[i]=y[j];naz[i]=z[j];
i=1;j=3;
nax[i]=x[j];nay[i]=y[j];naz[i]=z[j];
i=2;j=1;
nax[i]=x[j];nay[i]=y[j];naz[i]=z[j];
i=3;j=2;
nax[i]=x[j];nay[i]=y[j];naz[i]=z[j];

if(1) mz=RESO-1-ds2;

i=4;j=0;
nax[i]=x[j];nay[i]=y[j];naz[i]=mz;
i=5;j=3;
```

```
nax[i]=x[j];nay[i]=y[j];naz[i]=mz;
i=6;j=1;
nax[i]=x[j];nay[i]=y[j];naz[i]=mz;
i=7;j=2;
nax[i]=x[j];nay[i]=y[j];naz[i]=mz;
}
}/** set_seedpoints_ **/


int CW_pl(int pos,int jmax)
{
int j,count,val[2];
int old;

count=0;
for(j=pos;;){
/* here */
val[0]=rpixel(fen("X",j,jmax),fen("Y",j,jmax),fen("Z",j,jmax));
val[1]=rpixel(fen("X",j/*-1*/,jmax),fen("Y",j/*-1*/,jmax),fen("Z",j/*-1*/,jmax));

if(/*val[0]!=ca && *//*val[1]*/cc[j]==ca && !in[j]){
/* here */
X=fen("X",j/*-1*/,jmax);Y=fen("Y",j/*-1*/,jmax);Z=fen("Z",j/*-1*/,jmax);
X_=fen("X_",j/*-1*/,jmax);Y_=fen("Y_",j/*-1*/,jmax);Z_=fen("Z_",j/*-1*/,jmax);

return 0;
}

j--;if(j<0) j=jmax;
count++;
if(count==jmax+1) {printf(" ?CW_pl\n");refill=0;return 1;}
}/**for()**/

}/** CW_pl **/


int CCW_pl(int pos,int jmax)
{
int j,count,val[2];
int old;

count=0;
for(j=pos;;){
/* here */
val[0]=rpixel(fen("X",j,jmax),fen("Y",j,jmax),fen("Z",j,jmax));
val[1]=rpixel(fen("X",j/*+1*/,jmax),fen("Y",j/*+1*/,jmax),fen("Z",j/*+1*/,jmax));

if(/*val[0]!=ca && *//*val[1]*/cc[j]==ca && !in[j]){
/* here */
X=fen("X",j/*+1*/,jmax);Y=fen("Y",j/*+1*/,jmax);Z=fen("Z",j/*+1*/,jmax);
X_=fen("X_",j/*+1*/,jmax);Y_=fen("Y_",j/*+1*/,jmax);Z_=fen("Z_",j/*+1*/,jmax);

return 0;
}
```

```
j++;if(j>jmax) j=0;
count++;
if(count==jmax+1) {printf(" ?CCW_pl\n");refill=0;return 1;}
}/**for()**/

}/** CCW_pl **/


int CW(int pos,int jmax)
{
int j,count;

count=0;
for(j=pos/*+1*/;;){
/*if(j==pos && cc[j]==ca) refill=0;*/

if(/*cc[j]!=ca && */cc[jen(j-1,jmax)]==ca){
/* here */
X=fen("X",j-1,jmax);Y=fen("Y",j-1,jmax);Z=fen("Z",j-1,jmax);
X_=fen("X_",j-1,jmax);Y_=fen("Y_",j-1,jmax);Z_=fen("Z_",j-1,jmax);

return 0;
}

j--;if(j<0) j=jmax;
count++;
if(count==jmax+1) {printf(" ?CW\n");return 1;}
}/**for()**/
}/** CW **/


int CCW(int pos,int jmax)
{
int j,count;

count=0;
for(j=pos/*-1*/;;){
/*if(j==pos && cc[j]==ca) refill=0;*/

if(/*cc[j]!=ca && */cc[jen(j+1,jmax)]==ca){
/* here */
X=fen("X",j+1,jmax);Y=fen("Y",j+1,jmax);Z=fen("Z",j+1,jmax);
X_=fen("X_",j+1,jmax);Y_=fen("Y_",j+1,jmax);Z_=fen("Z_",j+1,jmax);

return 0;
}

j++;if(j>jmax) j=0;
count++;
if(count==jmax+1) {printf(" ?CCW\n");return 1;}
}/**for()**/
}/** CCW **/


int en_xy_p(void)
```

```
{
int j;

j=-1;
plflag=1;

if(c18>=wall){
j++;  /* ca18 */
    enX[j]=x[18];enY[j]=y[18];enZ[j]=z[18];
    enX_[j]=x_[18];enY_[j]=y_[18];enZ_[j]=z_[18];in[j]=0;cc[j]=c18;
}
if(c11>=wall){
j++;  /* ca11 */
    enX[j]=x[11];enY[j]=y[11];enZ[j]=z[11];
    enX_[j]=x_[11];enY_[j]=y_[11];enZ_[j]=z_[11];in[j]=1;cc[j]=c11;
}
if(c15>=wall){
j++;  /* ca15 */
    enX[j]=x[15];enY[j]=y[15];enZ[j]=z[15];
    enX_[j]=x_[15];enY_[j]=y_[15];enZ_[j]=z_[15];in[j]=0;cc[j]=c15;
}
if(c12>=wall){
j++;  /* ca12 */
    enX[j]=x[12];enY[j]=y[12];enZ[j]=z[12];
    enX_[j]=x_[12];enY_[j]=y_[12];enZ_[j]=z_[12];in[j]=1;cc[j]=c12;
}
if(c16>=wall){
j++;  /* ca16 */
    enX[j]=x[16];enY[j]=y[16];enZ[j]=z[16];
    enX_[j]=x_[16];enY_[j]=y_[16];enZ_[j]=z_[16];in[j]=0;cc[j]=c16;
}
if(c13>=wall){
j++;  /* ca13 */
    enX[j]=x[13];enY[j]=y[13];enZ[j]=z[13];
    enX_[j]=x_[13];enY_[j]=y_[13];enZ_[j]=z_[13];in[j]=1;cc[j]=c13;
}
if(c17>=wall){
j++;  /* ca17 */
    enX[j]=x[17];enY[j]=y[17];enZ[j]=z[17];
    enX_[j]=x_[17];enY_[j]=y_[17];enZ_[j]=z_[17];in[j]=0;cc[j]=c17;
}
if(c14>=wall){
j++;  /* ca14 */
    enX[j]=x[14];enY[j]=y[14];enZ[j]=z[14];
    enX_[j]=x_[14];enY_[j]=y_[14];enZ_[j]=z_[14];in[j]=1;cc[j]=c14;
}

return j;
}/** en_xy_p **/


int en_xy_m(void)
{
int j;
```

```
j=-1;
plflag=1;

if(c26>=wall){
j++;  /* ca26 */
    enX[j]=x[26];enY[j]=y[26];enZ[j]=z[26];
    enX_[j]=x_[26];enY_[j]=y_[26];enZ_[j]=z_[26];in[j]=0;cc[j]=c26;
}
if(c19>=wall){
j++;  /* ca19 */
    enX[j]=x[19];enY[j]=y[19];enZ[j]=z[19];
    enX_[j]=x_[19];enY_[j]=y_[19];enZ_[j]=z_[19];in[j]=1;cc[j]=c19;
}
if(c23>=wall){
j++;  /* ca23 */
    enX[j]=x[23];enY[j]=y[23];enZ[j]=z[23];
    enX_[j]=x_[23];enY_[j]=y_[23];enZ_[j]=z_[23];in[j]=0;cc[j]=c23;
}
if(c20>=wall){
j++;  /* ca20 */
    enX[j]=x[20];enY[j]=y[20];enZ[j]=z[20];
    enX_[j]=x_[20];enY_[j]=y_[20];enZ_[j]=z_[20];in[j]=1;cc[j]=c20;
}
if(c24>=wall){
j++;  /* ca24 */
    enX[j]=x[24];enY[j]=y[24];enZ[j]=z[24];
    enX_[j]=x_[24];enY_[j]=y_[24];enZ_[j]=z_[24];in[j]=0;cc[j]=c24;
}
if(c21>=wall){
j++;  /* ca21 */
    enX[j]=x[21];enY[j]=y[21];enZ[j]=z[21];
    enX_[j]=x_[21];enY_[j]=y_[21];enZ_[j]=z_[21];in[j]=1;cc[j]=c21;
}
if(c25>=wall){
j++;  /* ca25 */
    enX[j]=x[25];enY[j]=y[25];enZ[j]=z[25];
    enX_[j]=x_[25];enY_[j]=y_[25];enZ_[j]=z_[25];in[j]=0;cc[j]=c25;
}
if(c22>=wall){
j++;  /* ca22 */
    enX[j]=x[22];enY[j]=y[22];enZ[j]=z[22];
    enX_[j]=x_[22];enY_[j]=y_[22];enZ_[j]=z_[22];in[j]=1;cc[j]=c22;
}

return j;
}/** en_xy_m **/


int en_yz_p(void)
{
int j;

j=-1;
plflag=2;
```

```
if(c23>=wall){
j++;  /* ca23 */
    enX[j]=x[23];enY[j]=y[23];enZ[j]=z[23];
    enX_[j]=x_[23];enY_[j]=y_[23];enZ_[j]=z_[23];in[j]=0;cc[j]=c23;
}
if(c5>=wall){
j++;  /* ca5 */
    enX[j]=x[5];enY[j]=y[5];enZ[j]=z[5];
    enX_[j]=x_[5];enY_[j]=y_[5];enZ_[j]=z_[5];in[j]=1;cc[j]=c5;
}
if(c15>=wall){
j++;  /* ca15 */
    enX[j]=x[15];enY[j]=y[15];enZ[j]=z[15];
    enX_[j]=x_[15];enY_[j]=y_[15];enZ_[j]=z_[15];in[j]=0;cc[j]=c15;
}
if(c11>=wall){
j++;  /* ca11 */
    enX[j]=x[11];enY[j]=y[11];enZ[j]=z[11];
    enX_[j]=x_[11];enY_[j]=y_[11];enZ_[j]=z_[11];in[j]=1;cc[j]=c11;
}
if(c18>=wall){
j++;  /* ca18 */
    enX[j]=x[18];enY[j]=y[18];enZ[j]=z[18];
    enX_[j]=x_[18];enY_[j]=y_[18];enZ_[j]=z_[18];in[j]=0;cc[j]=c18;
}
if(c8>=wall){
j++;  /* ca8 */
    enX[j]=x[8];enY[j]=y[8];enZ[j]=z[8];
    enX_[j]=x_[8];enY_[j]=y_[8];enZ_[j]=z_[8];in[j]=1;cc[j]=c8;
}
if(c26>=wall){
j++;  /* ca26 */
    enX[j]=x[26];enY[j]=y[26];enZ[j]=z[26];
    enX_[j]=x_[26];enY_[j]=y_[26];enZ_[j]=z_[26];in[j]=0;cc[j]=c26;
}
if(c19>=wall){
j++;  /* ca19 */
    enX[j]=x[19];enY[j]=y[19];enZ[j]=z[19];
    enX_[j]=x_[19];enY_[j]=y_[19];enZ_[j]=z_[19];in[j]=1;cc[j]=c19;
}

return j;
}/** en_yz_p **/


int en_yz_m(void)
{
int j;

j=-1;
plflag=2;

if(c24>=wall){
j++;  /* ca24 */
    enX[j]=x[24];enY[j]=y[24];enZ[j]=z[24];
```

```
    enX_[j]=x_[24];enY_[j]=y_[24];enZ_[j]=z_[24];in[j]=0;cc[j]=c24;
}
if(c6>=wall){
j++;  /* ca6 */
    enX[j]=x[6];enY[j]=y[6];enZ[j]=z[6];
    enX_[j]=x_[6];enY_[j]=y_[6];enZ_[j]=z_[6];in[j]=1;cc[j]=c6;
}
if(c16>=wall){
j++;  /* ca16 */
    enX[j]=x[16];enY[j]=y[16];enZ[j]=z[16];
    enX_[j]=x_[16];enY_[j]=y_[16];enZ_[j]=z_[16];in[j]=0;cc[j]=c16;
}
if(c13>=wall){
j++;  /* ca13 */
    enX[j]=x[13];enY[j]=y[13];enZ[j]=z[13];
    enX_[j]=x_[13];enY_[j]=y_[13];enZ_[j]=z_[13];in[j]=1;cc[j]=c13;
}
if(c17>=wall){
j++;  /* ca17 */
    enX[j]=x[17];enY[j]=y[17];enZ[j]=z[17];
    enX_[j]=x_[17];enY_[j]=y_[17];enZ_[j]=z_[17];in[j]=0;cc[j]=c17;
}
if(c7>=wall){
j++;  /* ca7 */
    enX[j]=x[7];enY[j]=y[7];enZ[j]=z[7];
    enX_[j]=x_[7];enY_[j]=y_[7];enZ_[j]=z_[7];in[j]=1;cc[j]=c7;
}
if(c25>=wall){
j++;  /* ca25 */
    enX[j]=x[25];enY[j]=y[25];enZ[j]=z[25];
    enX_[j]=x_[25];enY_[j]=y_[25];enZ_[j]=z_[25];in[j]=0;cc[j]=c25;
}
if(c21>=wall){
j++;  /* ca21 */
    enX[j]=x[21];enY[j]=y[21];enZ[j]=z[21];
    enX_[j]=x_[21];enY_[j]=y_[21];enZ_[j]=z_[21];in[j]=1;cc[j]=c21;
}

return j;
}/** en_yz_m **/


int en_zx_p(void)
{
int j;

j=-1;
plflag=3;

if(c16>=wall){
j++;  /* ca16 */
    enX[j]=x[16];enY[j]=y[16];enZ[j]=z[16];
    enX_[j]=x_[16];enY_[j]=y_[16];enZ_[j]=z_[16];in[j]=0;cc[j]=c16;
}
if(c12>=wall){
```

```
j++;  /* ca12 */
    enX[j]=x[12];enY[j]=y[12];enZ[j]=z[12];
    enX_[j]=x_[12];enY_[j]=y_[12];enZ_[j]=z_[12];in[j]=1;cc[j]=c12;
}
if(c15>=wall){
j++;  /* ca15 */
    enX[j]=x[15];enY[j]=y[15];enZ[j]=z[15];
    enX_[j]=x_[15];enY_[j]=y_[15];enZ_[j]=z_[15];in[j]=0;cc[j]=c15;
}
if(c5>=wall){
j++;  /* ca5 */
    enX[j]=x[5];enY[j]=y[5];enZ[j]=z[5];
    enX_[j]=x_[5];enY_[j]=y_[5];enZ_[j]=z_[5];in[j]=1;cc[j]=c5;
}
if(c23>=wall){
j++;  /* ca23 */
    enX[j]=x[23];enY[j]=y[23];enZ[j]=z[23];
    enX_[j]=x_[23];enY_[j]=y_[23];enZ_[j]=z_[23];in[j]=0;cc[j]=c23;
}
if(c20>=wall){
j++;  /* ca20 */
    enX[j]=x[20];enY[j]=y[20];enZ[j]=z[20];
    enX_[j]=x_[20];enY_[j]=y_[20];enZ_[j]=z_[20];in[j]=1;cc[j]=c20;
}
if(c24>=wall){
j++;  /* ca24 */
    enX[j]=x[24];enY[j]=y[24];enZ[j]=z[24];
    enX_[j]=x_[24];enY_[j]=y_[24];enZ_[j]=z_[24];in[j]=0;cc[j]=c24;
}
if(c6>=wall){
j++;  /* ca6 */
    enX[j]=x[6];enY[j]=y[6];enZ[j]=z[6];
    enX_[j]=x_[6];enY_[j]=y_[6];enZ_[j]=z_[6];in[j]=1;cc[j]=c6;
}

return j;
}/** en_zx_p **/


int en_zx_m(void)
{
int j;

j=-1;
plflag=3;

if(c17>=wall){
j++;  /* ca17 */
    enX[j]=x[17];enY[j]=y[17];enZ[j]=z[17];
    enX_[j]=x_[17];enY_[j]=y_[17];enZ_[j]=z_[17];in[j]=0;cc[j]=c17;
}
if(c14>=wall){
j++;  /* ca14 */
    enX[j]=x[14];enY[j]=y[14];enZ[j]=z[14];
    enX_[j]=x_[14];enY_[j]=y_[14];enZ_[j]=z_[14];in[j]=1;cc[j]=c14;
```

```
}
if(c18>=wall){
j++;  /* ca18 */
    enX[j]=x[18];enY[j]=y[18];enZ[j]=z[18];
    enX_[j]=x_[18];enY_[j]=y_[18];enZ_[j]=z_[18];in[j]=0;cc[j]=c18;
}
if(c8>=wall){
j++;  /* ca8 */
    enX[j]=x[8];enY[j]=y[8];enZ[j]=z[8];
    enX_[j]=x_[8];enY_[j]=y_[8];enZ_[j]=z_[8];in[j]=1;cc[j]=c8;
}
if(c26>=wall){
j++;  /* ca26 */
    enX[j]=x[26];enY[j]=y[26];enZ[j]=z[26];
    enX_[j]=x_[26];enY_[j]=y_[26];enZ_[j]=z_[26];in[j]=0;cc[j]=c26;
}
if(c22>=wall){
j++;  /* ca22 */
    enX[j]=x[22];enY[j]=y[22];enZ[j]=z[22];
    enX_[j]=x_[22];enY_[j]=y_[22];enZ_[j]=z_[22];in[j]=1;cc[j]=c22;
}
if(c25>=wall){
j++;  /* ca25 */
    enX[j]=x[25];enY[j]=y[25];enZ[j]=z[25];
    enX_[j]=x_[25];enY_[j]=y_[25];enZ_[j]=z_[25];in[j]=0;cc[j]=c25;
}
if(c7>=wall){
j++;  /* ca7 */
    enX[j]=x[7];enY[j]=y[7];enZ[j]=z[7];
    enX_[j]=x_[7];enY_[j]=y_[7];enZ_[j]=z_[7];in[j]=1;cc[j]=c7;
}

return j;
}/** en_zx_m **/


int en_xy(void)
{
int j;

j=-1;

if(c1>=wall){
j++;  /* ca1 */
    enX[j]=x[1];enY[j]=y[1];enZ[j]=z[1];
    enX_[j]=x_[1];enY_[j]=y_[1];enZ_[j]=z_[1];cc[j]=c1;
}
if(c4>=wall){
j++;  /* ca4 */
    enX[j]=x[4];enY[j]=y[4];enZ[j]=z[4];
    enX_[j]=x_[4];enY_[j]=y_[4];enZ_[j]=z_[4];cc[j]=c4;
}
if(c3>=wall){
j++;  /* ca3 */
    enX[j]=x[3];enY[j]=y[3];enZ[j]=z[3];
```

```
    enX_[j]=x_[3];enY_[j]=y_[3];enZ_[j]=z_[3];cc[j]=c3;
}
if(c2>=wall){
j++;  /* ca2 */
    enX[j]=x[2];enY[j]=y[2];enZ[j]=z[2];
    enX_[j]=x_[2];enY_[j]=y_[2];enZ_[j]=z_[2];cc[j]=c2;
}

return j;
}/** en_xy **/


int en_yz(void)
{
int j;

j=-1;

if(c2>=wall){
j++;  /* ca2 */
    enX[j]=x[2];enY[j]=y[2];enZ[j]=z[2];
    enX_[j]=x_[2];enY_[j]=y_[2];enZ_[j]=z_[2];cc[j]=c2;
}
if(c10>=wall){
j++;  /* ca10 */
    enX[j]=x[10];enY[j]=y[10];enZ[j]=z[10];
    enX_[j]=x_[10];enY_[j]=y_[10];enZ_[j]=z_[10];cc[j]=c10;
}
if(c4>=wall){
j++;  /* ca4 */
    enX[j]=x[4];enY[j]=y[4];enZ[j]=z[4];
    enX_[j]=x_[4];enY_[j]=y_[4];enZ_[j]=z_[4];cc[j]=c4;
}
if(c9>=wall){
j++;  /* ca9 */
    enX[j]=x[9];enY[j]=y[9];enZ[j]=z[9];
    enX_[j]=x_[9];enY_[j]=y_[9];enZ_[j]=z_[9];cc[j]=c9;
}

return j;
}/** en_yz **/


int en_zx(void)
{
int j;

j=-1;

if(c9>=wall){
j++;  /* ca9 */
    enX[j]=x[9];enY[j]=y[9];enZ[j]=z[9];
    enX_[j]=x_[9];enY_[j]=y_[9];enZ_[j]=z_[9];cc[j]=c9;
}
if(c3>=wall){
```

```
j++;  /* ca3 */
    enX[j]=x[3];enY[j]=y[3];enZ[j]=z[3];
    enX_[j]=x_[3];enY_[j]=y_[3];enZ_[j]=z_[3];cc[j]=c3;
}
if(c10>=wall){
j++;  /* ca10 */
    enX[j]=x[10];enY[j]=y[10];enZ[j]=z[10];
    enX_[j]=x_[10];enY_[j]=y_[10];enZ_[j]=z_[10];cc[j]=c10;
}
if(c1>=wall){
j++;  /* ca1 */
    enX[j]=x[1];enY[j]=y[1];enZ[j]=z[1];
    enX_[j]=x_[1];enY_[j]=y_[1];enZ_[j]=z_[1];cc[j]=c1;
}

return j;
}/** en_zx **/


int Su(int n1_,int n1)
{
/* here */
if(n1_==n1)
     return 1;
else return 0;
}/** Su **/


int C(int c1,int c2,int c3,int c4)
{
if(c1==ca || c2==ca || c3==ca || c4==ca) return 1;
else return 0;
}/** C **/


int cag_r(long oldtime)
{
char str[32],buf[10];
int i,j,jmax,k;
int flag_[CPMAX],flag_pp[CPMAX],S_old[3],NX[2],NY[2],NZ[2];
int cp,ssize,pos,cflag,algo;
int nxp,nxm,nyp,nym,nzp,nzm,nxp_c,nxm_c,nyp_c,nym_c,nzp_c,nzm_c;
long pl4count;
dbl val;

if(refill==0) return 1;

cnt++;
ssize=sizeof(sss);
if(c_trans==0 || SPmode==1) cp=CPMAX;else cp=1;

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

```
pl4count=0;

for(i=0;i<CPMAX;i++){
strcpy(str,"rtn[");
#if !WX
strcat(str,itoa(i,buf,10));
#else
strcat(str,gcvt(i,8,buf));
#endif
strcat(str,"].bin");

if((fp[i]=FOPEN(str,"w+b"))==NULL){
strcpy(str,"Do \"ulimit -n ");
#if !WX
strcat(str,itoa(CPMAX+1000,buf,10));
#else
strcat(str,gcvt(CPMAX+1000,8,buf));
#endif
strcat(str,"\".");
printf(" %s\n",str);
refill=0;
return 1;
}
}

ca=/*16*/fcolor;
if(CPMAX==CPMAXh*2){
for(j=0;j<CPMAX;j++){
if(j<CPMAXh) pl[j]=0+0;
else{
if(K_) pl[j]=K_;                    /* auto */
else   pl[j]=/*1*//*2*//*3*/3+0;    /* manual */
}
}
}
else if(CPMAX==CPMAXh*4){
for(j=0;j<CPMAX;j++) pl[j]=j/CPMAXh;
}
else{
for(j=0;j<CPMAX;j++) pl[j]=0;
}

if(c_trans==0 || SPmode==1){
set_colors();
if(CPMAX==1){
/* here */
nax[0]=1;nay[0]=1;naz[0]=1;
}
else if(!PBC) set_seedpoints_();
else          set_seedpoints();
}
else{
}

if(CPMAX>=CPMAXh*2){
```

```
for(j=CPMAXh;j<CPMAXh*2;j++){
nax[j]=nax[j-CPMAXh];
nay[j]=nay[j-CPMAXh];
naz[j]=naz[j-CPMAXh];
}
}

if(CPMAX==CPMAXh*4){
for(j=CPMAXh*2;j<CPMAX;j++){
nax[j]=nax[j-CPMAXh*2];
nay[j]=nay[j-CPMAXh*2];
naz[j]=naz[j-CPMAXh*2];
}
}

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

/* here */
nx[pl[i]][i]=nax[i];ny[pl[i]][i]=nay[i];nz[pl[i]][i]=naz[i];
/* here */
putpixel_(nx[pl[i]][i],ny[pl[i]][i],nz[pl[i]][i],acolor[i]);
pcount[i]++;
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX) break;}
else break;
}/**while(1)**/

i=0;
while(1){
if(flag_[i]){                        /* CP_? */
/* here */
nx_[pl[i]][i]=nax[i];ny_[pl[i]][i]=nay[i];nz_[pl[i]][i]=naz[i];
plx_[i]=nx_[pl[i]][i];ply_[i]=ny_[pl[i]][i];plz_[i]=nz_[pl[i]][i];
ig=i;

if(c_trans==0 || SPmode==1){
if(PBC || CPMAX==1){
if(CPMAX==CPMAXh*2 && /*i<CPMAXh*/(i/CPMAXh)%2==0) nax[i]--;
else                             nax[i]++;
}
else{
if(CPMAX==2){
if(i%2==0) nax[i]++;
else if(i%2==1) nax[i]--;
/*if(i%2==0) nay[i]++;
else if(i%2==1) nay[i]--;*/
}
else if(CPMAX==4){
/* lower, upper */
/* lower */
```

```
if(1){
if(i%4==0) nax[i]++;
else if(i%4==1) nax[i]--;

else if(i%4==2) nay[i]--;
else if(i%4==3) nay[i]++;
/*else if(i==2) nax[i]++;
else if(i==3) nax[i]--;*/
}
else{
if(i%4==0) naz[i]++;
else if(i%4==1) naz[i]--;

else if(i%4==2) nax[i]--;
else if(i%4==3) nax[i]++;
}
}/**else if(CPMAX)**/
else if(CPMAX==8){
/*if(i<CPMAX/2){
if(i%4==0) nax[i]++;
else if(i%4==1) nax[i]--;

else if(i%4==2) nay[i]--;
else if(i%4==3) nay[i]++;
}
else{
if(i%4==0) nay[i]++;
else if(i%4==1) nay[i]--;

else if(i%4==2) nax[i]++;
else if(i%4==3) nax[i]--;
}*/
if(i<CPMAX/2){
if(i%4==0) nay[i]++;
else if(i%4==1) nay[i]--;

else if(i%4==2) nax[i]++;
else if(i%4==3) nax[i]--;
}
else{
if(i%4==0) nax[i]++;
else if(i%4==1) nax[i]--;

else if(i%4==2) nay[i]--;
else if(i%4==3) nay[i]++;
}
}/**else if(CPMAX)**/
}/**else(PBC,CPMAX)**/
}
else{
}

/* here */
nx[pl[i]][i]=nax[i];ny[pl[i]][i]=nay[i];nz[pl[i]][i]=naz[i];
```

```
jump(i);                /* modification of b,S */

/* here */
putpixel_(nx[pl[i]][i],ny[pl[i]][i],nz[pl[i]][i],acolor[i]);
pcount[i]++;
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX) break;}
else break;
}/**while(1)**/

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

while(cp){
kbhit_();
/*if(CPMAX==CPMAXh*2 && _3d14 && K_ && pl[0]==pl[CPMAXh]) refill=0;*/
if(refill<=0) break;

algo=0+random_(2);

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

/* here **/
if(PBC){
}
else{
}

/* here */
Nx=nx[pl[i]][i];Ny=ny[pl[i]][i];Nz=nz[pl[i]][i];
/*Nx_=nx_[pl[i]][i];Ny_=ny_[pl[i]][i];Nz_=nz_[pl[i]][i];*/
Nx_=plx_[i];Ny_=ply_[i];Nz_=plz_[i];  /* jumped */
S_old[0]=Nx-Nx_;S_old[1]=Ny-Ny_;S_old[2]=Nz-Nz_;
nxp=nx[pl[i]][i]+1;nyp=ny[pl[i]][i]+1;nzp=nz[pl[i]][i]+1;
nxm=nx[pl[i]][i]-1;nym=ny[pl[i]][i]-1;nzm=nz[pl[i]][i]-1;
nxp_c=connect_nxpm(nxp);nxm_c=connect_nxpm(nxm);
nyp_c=connect_nypm(nyp);nym_c=connect_nypm(nym);
nzp_c=connect_nzpm(nzp);nzm_c=connect_nzpm(nzm);

if((val=sqrt(pow(Nx-Nx_,2)+pow(Ny-Ny_,2)+pow(Nz-Nz_,2)))>1){
printf(" ?len:%f\n",val);refill=0;
}

/* here */
c1=getpixel_(nxp_c,ny[pl[i]][i],nz[pl[i]][i]);
j=1;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;
c2=getpixel_(nx[pl[i]][i],nyp_c,nz[pl[i]][i]);
j=2;
```

```
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;
c3=getpixel_(nxm_c,ny[pl[i]][i],nz[pl[i]][i]);
j=3;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;
c4=getpixel_(nx[pl[i]][i],nym_c,nz[pl[i]][i]);
j=4;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c9=getpixel_(nx[pl[i]][i],ny[pl[i]][i],nzp_c);
j=9;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;
c10=getpixel_(nx[pl[i]][i],ny[pl[i]][i],nzm_c);
j=10;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;



c11=getpixel_pl(/*nxp*/1,/*ny[pl[i]][i]*/0,/*nzp*/1);
j=11;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c19=getpixel_pl(/*nxp*/1,/*ny[pl[i]][i]*/0,/*nzm*/-1);
j=19;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c21=getpixel_pl(/*nxm*/-1,/*ny[pl[i]][i]*/0,/*nzm*/-1);
j=21;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c13=getpixel_pl(/*nxm*/-1,/*ny[pl[i]][i]*/0,/*nzp*/1);
j=13;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;



c12=getpixel_pl(/*nx[pl[i]][i]*/0,/*nyp*/1,/*nzp*/1);
j=12;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c14=getpixel_pl(/*nx[pl[i]][i]*/0,/*nyp*/-1,/*nzp*/1);
j=14;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c22=getpixel_pl(/*nx[pl[i]][i]*/0,/*nyp*/-1,/*nzp*/-1);
j=22;
x[j]=X;y[j]=Y;z[j]=Z;
```

```
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c20=getpixel_pl(/*nx[pl[i]][i]*/0,/*nyp*/1,/*nzp*/-1);
j=20;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;


c5=getpixel_pl(/*nxp*/1,/*nyp*/1,/*nz[pl[i]][i]*/0);
j=5;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c6=getpixel_pl(/*nxp*/-1,/*nyp*/1,/*nz[pl[i]][i]*/0);
j=6;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c7=getpixel_pl(/*nxp*/-1,/*nyp*/-1,/*nz[pl[i]][i]*/0);
j=7;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;

c8=getpixel_pl(/*nxp*/1,/*nyp*/-1,/*nz[pl[i]][i]*/0);
j=8;
x[j]=X;y[j]=Y;z[j]=Z;
x_[j]=X_;y_[j]=Y_;z_[j]=Z_;


#if _3d14==-2
     if(pl[i]==1) {c12=c14=c22=c20=-2;c5=c6=c7=c8=-2;}
else if(pl[i]==2) {c11=c19=c21=c13=-2;c5=c6=c7=c8=-2;}
else if(pl[i]==3) {c11=c19=c21=c13=-2;c12=c14=c22=c20=-2;}
#elif _3d14==1 || _3d14==-1
if(CPMAX<=CPMAXh*2){                /* auto */
if(K_>0){
     if(K_==1) {c12=c14=c22=c20=0;c5=c6=c7=c8=0;}
else if(K_==2) {c11=c19=c21=c13=0;c5=c6=c7=c8=0;}
else if(K_==3) {c11=c19=c21=c13=0;c12=c14=c22=c20=0;}
}
else if(K_==0){                     /* manual */
     if(0) {c12=c14=c22=c20=0;c5=c6=c7=c8=0;}
else if(0) {c11=c19=c21=c13=0;c5=c6=c7=c8=0;}
else if(1) {c11=c19=c21=c13=0;c12=c14=c22=c20=0;}
}
}
#endif




/* here */
if(((c1==ca)||(c2==ca)||(c3==ca)||(c4==ca)||(c9==ca)||(c10==ca))
   ||((c11==ca)||(c19==ca)||(c21==ca)||(c13==ca))
   ||((c12==ca)||(c14==ca)||(c22==ca)||(c20==ca))
```

```
   ||((c5==ca)||(c6==ca)||(c7==ca)||(c8==ca))
  ) cflag=1;
else cflag=0;

if(cflag){
/* here */
ss.pl=pl[i];
ss.xx=nx[pl[i]][i];ss.yy=ny[pl[i]][i];ss.zz=nz[pl[i]][i];
ss.xx_=nx_[pl[i]][i];ss.yy_=ny_[pl[i]][i];ss.zz_=nz_[pl[i]][i];
if(fwrite(&ss,1,ssize,fp[i])<ssize) {printf(" fwrite %ld\n",pcount[0]);refill=0;}

plflag=0;

if(algo/2==0){
if(C(c1,c2,c3,c4)&& Su(Nz_,Nz)){
jmax=en_xy();                       /* xy */
}
else if(C(c2,c4,c9,c10)&& Su(Nx_,Nx)){
jmax=en_yz();                       /* yz */
}
else if(C(c1,c3,c9,c10)&& Su(Ny_,Ny)){
jmax=en_zx();                       /* zx */
}



#if 0
#else
else{
plflag=-1;
if(/*i<CPMAXh*/(i/CPMAXh)%2==0){
/*999*/
if(C(c11,c19,c21,c13)){
if(pl[i]==0) pl[i]=1;
else if(pl[i]==1) pl[i]=0;
else if(pl[i]==2) pl[i]=3;
else if(pl[i]==3) pl[i]=2;
     if(c11==ca) {nx[pl[i]][i]=x[11];ny[pl[i]][i]=y[11];nz[pl[i]][i]=z[11];}
else if(c19==ca) {nx[pl[i]][i]=x[19];ny[pl[i]][i]=y[19];nz[pl[i]][i]=z[19];}
else if(c21==ca) {nx[pl[i]][i]=x[21];ny[pl[i]][i]=y[21];nz[pl[i]][i]=z[21];}
else if(c13==ca) {nx[pl[i]][i]=x[13];ny[pl[i]][i]=y[13];nz[pl[i]][i]=z[13];}
}
else if(C(c12,c14,c22,c20)){
if(pl[i]==0) pl[i]=2;
else if(pl[i]==1) pl[i]=3;
else if(pl[i]==2) pl[i]=0;
else if(pl[i]==3) pl[i]=1;
     if(c12==ca) {nx[pl[i]][i]=x[12];ny[pl[i]][i]=y[12];nz[pl[i]][i]=z[12];}
else if(c14==ca) {nx[pl[i]][i]=x[14];ny[pl[i]][i]=y[14];nz[pl[i]][i]=z[14];}
else if(c22==ca) {nx[pl[i]][i]=x[22];ny[pl[i]][i]=y[22];nz[pl[i]][i]=z[22];}
else if(c20==ca) {nx[pl[i]][i]=x[20];ny[pl[i]][i]=y[20];nz[pl[i]][i]=z[20];}
}
else if(C(c5,c6,c7,c8)){
if(pl[i]==0) pl[i]=3;
else if(pl[i]==1) pl[i]=2;
else if(pl[i]==2) pl[i]=1;
```

```
else if(pl[i]==3) pl[i]=0;
     if(c5==ca) {nx[pl[i]][i]=x[5];ny[pl[i]][i]=y[5];nz[pl[i]][i]=z[5];}
else if(c6==ca) {nx[pl[i]][i]=x[6];ny[pl[i]][i]=y[6];nz[pl[i]][i]=z[6];}
else if(c7==ca) {nx[pl[i]][i]=x[7];ny[pl[i]][i]=y[7];nz[pl[i]][i]=z[7];}
else if(c8==ca) {nx[pl[i]][i]=x[8];ny[pl[i]][i]=y[8];nz[pl[i]][i]=z[8];}
}
else {printf(" ??algo/2\n");refill=0;}
}/**if(i,CPMAXh)**/
else{
if(C(c11,c19,c21,c13)){
if(pl[i]==0) pl[i]=1;
else if(pl[i]==1) pl[i]=0;
else if(pl[i]==2) pl[i]=3;
else if(pl[i]==3) pl[i]=2;
     if(c21==ca) {nx[pl[i]][i]=x[21];ny[pl[i]][i]=y[21];nz[pl[i]][i]=z[21];}
else if(c13==ca) {nx[pl[i]][i]=x[13];ny[pl[i]][i]=y[13];nz[pl[i]][i]=z[13];}
else if(c11==ca) {nx[pl[i]][i]=x[11];ny[pl[i]][i]=y[11];nz[pl[i]][i]=z[11];}
else if(c19==ca) {nx[pl[i]][i]=x[19];ny[pl[i]][i]=y[19];nz[pl[i]][i]=z[19];}
}
else if(C(c12,c14,c22,c20)){
if(pl[i]==0) pl[i]=2;
else if(pl[i]==1) pl[i]=3;
else if(pl[i]==2) pl[i]=0;
else if(pl[i]==3) pl[i]=1;
     if(c22==ca) {nx[pl[i]][i]=x[22];ny[pl[i]][i]=y[22];nz[pl[i]][i]=z[22];}
else if(c20==ca) {nx[pl[i]][i]=x[20];ny[pl[i]][i]=y[20];nz[pl[i]][i]=z[20];}
else if(c12==ca) {nx[pl[i]][i]=x[12];ny[pl[i]][i]=y[12];nz[pl[i]][i]=z[12];}
else if(c14==ca) {nx[pl[i]][i]=x[14];ny[pl[i]][i]=y[14];nz[pl[i]][i]=z[14];}
}
else if(C(c5,c6,c7,c8)){
if(pl[i]==0) pl[i]=3;
else if(pl[i]==1) pl[i]=2;
else if(pl[i]==2) pl[i]=1;
else if(pl[i]==3) pl[i]=0;
     if(c7==ca) {nx[pl[i]][i]=x[7];ny[pl[i]][i]=y[7];nz[pl[i]][i]=z[7];}
else if(c8==ca) {nx[pl[i]][i]=x[8];ny[pl[i]][i]=y[8];nz[pl[i]][i]=z[8];}
else if(c5==ca) {nx[pl[i]][i]=x[5];ny[pl[i]][i]=y[5];nz[pl[i]][i]=z[5];}
else if(c6==ca) {nx[pl[i]][i]=x[6];ny[pl[i]][i]=y[6];nz[pl[i]][i]=z[6];}
}
else {printf(" ???algo/2\n");refill=0;}
}/**else(i,CPMAXh)**/
}
#endif
}/**if(algo/2)**/



if(!plflag){
for(j=0;j<=jmax;j++){
/* here */                          /* en?:connected */
if(enX[j]==connect_nxpm(Nx_) && enY[j]==connect_nypm(Ny_) && enZ[j]==connect_nzpm(Nz_)){
pos=j;break;
}
}
if(j>jmax) printf(" ?j\n");
```

```
if(algo%2==0) CW(pos,jmax);
else          CCW(pos,jmax);
/* here */
nx[pl[i]][i]=X;ny[pl[i]][i]=Y;nz[pl[i]][i]=Z;
nx_[pl[i]][i]=Nx;ny_[pl[i]][i]=Ny;nz_[pl[i]][i]=Nz;
}/**if(!plflag)**/
else if(plflag>0){
for(j=0;j<=jmax;j++){               /* in c5,c6,...,c22 en?:not connected */
     if(plflag==1) {if(in[j] && enX[j]==Nx_ && enY[j]==Ny_) {pos=j;break;}}
else if(plflag==2) {if(in[j] && enY[j]==Ny_ && enZ[j]==Nz_) {pos=j;break;}}
else if(plflag==3) {if(in[j] && enX[j]==Nx_ && enZ[j]==Nz_) {pos=j;break;}}
}
if(j>jmax) {printf(" ? %d\n",plflag);refill=0;}

if(algo%2==0) CW_pl(pos,jmax);
else          CCW_pl(pos,jmax);

if(pl[i]==0) pl[i]=1;else pl[i]=0;  /* the other */

nx[pl[i]][i]=X;ny[pl[i]][i]=Y;nz[pl[i]][i]=Z;
if(1) {X_=X-S_old[0];Y_=Y-S_old[1];Z_=Z-S_old[2];}
else  {X_=X+S_old[0];Y_=Y+S_old[1];Z_=Z+S_old[2];}
nx_[pl[i]][i]=/*Nx*/X_;ny_[pl[i]][i]=/*Ny*/Y_;nz_[pl[i]][i]=/*Nz*/Z_;

pl4count++;
}/**else if(plflag)**/
else{
X=nx[pl[i]][i];Y=ny[pl[i]][i];Z=nz[pl[i]][i];
if(1) {X_=X-S_old[0];Y_=Y-S_old[1];Z_=Z-S_old[2];}
else  {X_=X+S_old[0];Y_=Y+S_old[1];Z_=Z+S_old[2];}
nx_[pl[i]][i]=X_;ny_[pl[i]][i]=Y_;nz_[pl[i]][i]=Z_;

pl4count++;
}/**else(plflag)**/

plx_[i]=nx_[pl[i]][i];ply_[i]=ny_[pl[i]][i];plz_[i]=nz_[pl[i]][i];
jump(i);                /* modification of b,S */

/* here */
/*pixel[pl[i]][X][Y][Z]=acolor[i];*/
putpixel_(nx[pl[i]][i],ny[pl[i]][i],nz[pl[i]][i],acolor[i]);
pcount[i]++;

#if 0
if(pcount[/*i*/CPMAX-1]==125/*250*/){
/*printf(" i:%2d pl:%d %d %d %d\n",i,pl[i],nx[pl[i]][i],ny[pl[i]][i],nz[pl[i]][i]);*/
for(k=0;k<CPMAXh;k++)
if((nx[pl[k]][k]+nx[pl[CPMAX-1-k]][CPMAX-1-k])!=xt
   || (ny[pl[k]][k]+ny[pl[CPMAX-1-k]][CPMAX-1-k])!=yt
   || (nz[pl[k]][k]+nz[pl[CPMAX-1-k]][CPMAX-1-k])!=zt) refill=0;
}
#endif

flag_pp[i]=1;
}/**if(cflag)**/
```

```
else{
plflag=-2;

if(/*ftell_mem(i)*/ftell(fp[i])==0) {flag_[i]=0;cp--;if(cp==0) break;/*goto pp;*/}
/*fread_mem(i);*/
fseek(fp[i],-ssize,SEEK_CUR);
if(fread(&ss,1,ssize,fp[i])<ssize) {printf(" fread %ld\n",pcount[0]);refill=0;}
/* here */
pl[i]=ss.pl;
nx[pl[i]][i]=ss.xx;ny[pl[i]][i]=ss.yy;nz[pl[i]][i]=ss.zz;
nx_[pl[i]][i]=ss.xx_;ny_[pl[i]][i]=ss.yy_;nz_[pl[i]][i]=ss.zz_;
plx_[i]=nx_[pl[i]][i];ply_[i]=ny_[pl[i]][i];plz_[i]=nz_[pl[i]][i];
jump(i);                /* modification of b,S */
fseek(fp[i],-ssize,SEEK_CUR);

pp:
flag_pp[i]=0;
}/**else(cflag)**/
}/**if(flag_[i])**/

i++;
if(c_trans==0 || SPmode==1) {if(i==CPMAX) break;}
else break;
}/**while(1)**/
/*if(pcount[0]==10) refill=0;*/
}/**while(cp)**/

for(i=0;i<CPMAX;i++) fclose(fp[i]);
printf(" pl4:%ld\n",pl4count);

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