|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

|  |  |  |
| --- | --- | --- |
| 8 | 7 | 6 |
| 5 | 4 | 3 |
| 2 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

原理图1：立方体表面展开图

红色虚线框表示垂直于你视野的平面（分前，中，后三个平面）从右往上转动

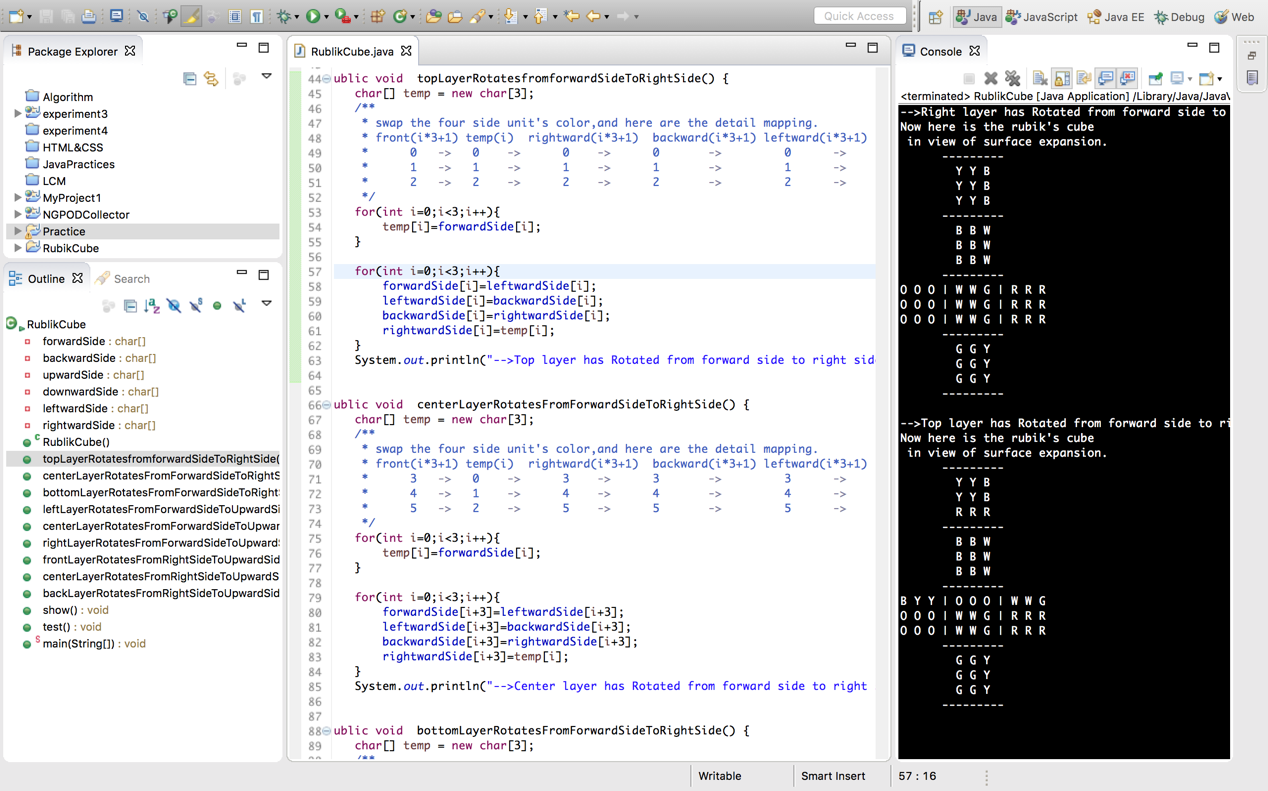
紫色虚线框表示平行于你视野的平面（分左，中，右三个平面）从前往上转动

绿色虚线框表示平行于你视野的平面（分上，中，下三个平面）从前往右转动

·

测试：

1.在魔方的每个原子变换行为的末尾，都添加了一行输出信息表示魔方进行了怎样的变换。方便调试，也使得过程形象客观。



收获：面在转动的时候（比如上面一层从前面的一侧向右转到右侧的时候），其实那个面也有变换。之前没有考虑到这一点。

/\*\*

\* **@author** tutianhuan

\* have a try to solve puzzle of the Rubik's Cube

\* and conclude the secret of rotation by computer

\*/

**public** **class** RublikCube {

**private** **char**[] forwardSide = **new** **char**[9];

**private** **char**[] backwardSide = **new** **char**[9];

**private** **char**[] upwardSide = **new** **char**[9];

**private** **char**[] downwardSide = **new** **char**[9];

**private** **char**[] leftwardSide = **new** **char**[9];

**private** **char**[] rightwardSide = **new** **char**[9];

/\*\*

\* construction function to initialize the state of cube,

\* and it behaves perfect if you don't input the original state.

\*/

**public** RublikCube(){

/\*\*

\* to initialize the foward side.

\* Originally it is blue upwards while green backwards,

\* orange on the left while red on the right,

\* and it is bule upward while green downwards.

\*/

**for**(**int** i=0;i<9;i++){

upwardSide[i]='B';

downwardSide[i]='G';

leftwardSide[i]='O';

rightwardSide[i]='R';

forwardSide[i]='W';

backwardSide[i]='Y';

}

}

**public** **void** topLayerRotatesfromforwardSideToRightSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the detail mapping.

\* front(i\*3+1) temp(i) rightward(i\*3+1) backward(i\*3+1) leftward(i\*3+1) front(i\*3+1)

\* 0 -> 0 -> 0 -> 0 -> 0 -> 0

\* 1 -> 1 -> 1 -> 1 -> 1 -> 1

\* 2 -> 2 -> 2 -> 2 -> 2 -> 2

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=forwardSide[i];

}

**for**(**int** i=0;i<3;i++){

forwardSide[i]=leftwardSide[i];

leftwardSide[i]=backwardSide[i];

backwardSide[i]=rightwardSide[i];

rightwardSide[i]=temp[i];

}

/\*\*

\* for upward side,it will be different as well.Here come the details.

\* i= 0 1 2 2 5 8 index=i\*3+2

\* i= 3 4 5 ===>> 1 4 7 index=(i-3)\*3+1=i\*3-8

\* i= 6 7 8 0 3 6 index=(i-6)\*3

\*/

**char**[] tempSide =**new** **char**[9];

**for**(**int** i=0;i<9;i++){

tempSide[i]=upwardSide[i];

}

**for**(**int** i=0;i<9;i++){

**if**(i<3)

upwardSide[i]=tempSide[i\*3+2];

**else** **if**(i<6)

upwardSide[i]=tempSide[(i-3)\*3+1];

**else**

upwardSide[i]=tempSide[(i-6)\*3];

}

System.***out***.println("-->Top layer has Rotated from forward side to right side.");

}

**public** **void** centerLayerRotatesFromForwardSideToRightSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the detail mapping.

\* front(i\*3+1) temp(i) rightward(i\*3+1) backward(i\*3+1) leftward(i\*3+1) front(i\*3+1)

\* 3 -> 0 -> 3 -> 3 -> 3 -> 3

\* 4 -> 1 -> 4 -> 4 -> 4 -> 4

\* 5 -> 2 -> 5 -> 5 -> 5 -> 5

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=forwardSide[i];

}

**for**(**int** i=0;i<3;i++){

forwardSide[i+3]=leftwardSide[i+3];

leftwardSide[i+3]=backwardSide[i+3];

backwardSide[i+3]=rightwardSide[i+3];

rightwardSide[i+3]=temp[i];

}

System.***out***.println("-->Center layer has Rotated from forward side to right side.");

}

**public** **void** bottomLayerRotatesFromForwardSideToRightSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the detail mapping.

\* front(i+6) temp(i+6) rightward(i+6) backward(i+6) leftward(i+6) front(i+6)

\* 6 -> 0 -> 6 -> 6 -> 6 -> 6

\* 7 -> 1 -> 7 -> 7 -> 7 -> 7

\* 8 -> 2 -> 8 -> 8 -> 8 -> 8

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=forwardSide[i+6];

}

**for**(**int** i=0;i<3;i++){

forwardSide[i+6]=leftwardSide[i+6];

leftwardSide[i+6]=backwardSide[i+6];

backwardSide[i+6]=rightwardSide[i+6];

rightwardSide[i+6]=temp[i];

}

/\*\*

\*for downward side,it will be different as well.Here come the details.

\*i= 0 1 2 6 3 0 index=6-i\*3

\*i= 3 4 5 ===>> 7 4 1 index=8-i

\*i= 6 7 8 8 5 2 index=8-(i-6)\*3=26-i\*3

\*/

**char**[] tempSide =**new** **char**[9];

**for**(**int** i=0;i<9;i++){

tempSide[i]=downwardSide[i];

}

**for**(**int** i=0;i<9;i++){

**if**(i<3)

downwardSide[i]=tempSide[6-i\*3];

**else** **if**(i<6)

downwardSide[i]=tempSide[8-i];

**else**

downwardSide[i]=tempSide[8-(i-6)\*3];

}

System.***out***.println("-->Bottom layer has Rotated from forward side to right side.");

}

**public** **void** leftLayerRotatesFromForwardSideToUpwardSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the details about mapping.

\* front(i\*3) temp(i) upward(i\*3) backward(8-i\*3) downward(i\*3) front(i\*3)

\* 0 -> 0 -> 0 -> 8 -> 0 -> 0

\* 3 -> 1 -> 3 -> 5 -> 3 -> 3

\* 6 -> 2 -> 6 -> 2 -> 6 -> 6

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=forwardSide[i\*3];

}

**for**(**int** i=0;i<3;i++){

forwardSide[i\*3]=downwardSide[i\*3];

downwardSide[i\*3]=backwardSide[8-i\*3];

backwardSide[8-i\*3]=upwardSide[i\*3];

upwardSide[i\*3]=temp[i];

}

/\*\*

\* for leftward side,it will be different as well.Here come the details.

\* i= 0 1 2 2 5 8 index=i\*3+2

\* i= 3 4 5 ===>> 1 4 7 index=(i-3)\*3+1=i\*3-8

\* i= 6 7 8 0 3 6 index=(i-6)\*3

\*/

**char**[] tempSide =**new** **char**[9];

**for**(**int** i=0;i<9;i++){

tempSide[i]=leftwardSide[i];

}

**for**(**int** i=0;i<9;i++){

**if**(i<3)

leftwardSide[i]=tempSide[i\*3+2];

**else** **if**(i<6)

leftwardSide[i]=tempSide[(i-3)\*3+1];

**else**

leftwardSide[i]=tempSide[(i-6)\*3];

}

System.***out***.println("-->Left layer has Rotated from forward side to upward side.");

}

**public** **void** centerLayerRotatesFromForwardSideToUpwardSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the details about mapping.

\* front(i\*3+1) temp(i) upward(i\*3+1) backward(i\*3+1) downward(i\*3+1) front(i\*3+1)

\* 1 -> 0 -> 1 -> 7 -> 1 -> 1

\* 4 -> 1 -> 4 -> 4 -> 4 -> 4

\* 7 -> 2 -> 7 -> 1 -> 7 -> 7

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=forwardSide[i\*3+1];

}

**for**(**int** i=0;i<3;i++){

forwardSide[i\*3+1]=downwardSide[i\*3+1];

downwardSide[i\*3+1]=backwardSide[8-(i\*3+1)];

backwardSide[8-(i\*3+1)]=upwardSide[i\*3+1];

upwardSide[i\*3+1]=temp[i];

}

System.***out***.println("-->Center layer has Rotated from forward side to upward side.");

}

**public** **void** rightLayerRotatesFromForwardSideToUpwardSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the details about mapping.

\* front(i\*3+2) temp(i) upward(i\*3+2) backward(6-i\*3) downward(i\*3+2) front(i\*3+2)

\* 2 -> 0 -> 2 -> 6 -> 2 -> 2

\* 5 -> 1 -> 5 -> 3 -> 5 -> 5

\* 8 -> 2 -> 8 -> 0 -> 8 -> 8

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=forwardSide[i\*3+2];

}

**for**(**int** i=0;i<3;i++){

forwardSide[i\*3+2]=downwardSide[i\*3+2];

downwardSide[i\*3+2]=backwardSide[6-i\*3];

backwardSide[6-i\*3]=upwardSide[i\*3+2];

upwardSide[i\*3+2]=temp[i];

}

/\*\*

\*for rightward side,it will be different as well.Here come the details.

\*i= 0 1 2 6 3 0 index=6-i\*3

\*i= 3 4 5 ===>> 7 4 1 index=8-i

\*i= 6 7 8 8 5 2 index=8-(i-6)\*3=26-i\*3

\*/

**char**[] tempSide =**new** **char**[9];

**for**(**int** i=0;i<9;i++){

tempSide[i]=rightwardSide[i];

}

**for**(**int** i=0;i<9;i++){

**if**(i<3)

rightwardSide[i]=tempSide[6-i\*3];

**else** **if**(i<6)

rightwardSide[i]=tempSide[8-i];

**else**

rightwardSide[i]=tempSide[8-(i-6)\*3];

}

System.***out***.println("-->Right layer has Rotated from forward side to upward side.");

}

**public** **void** frontLayerRotatesFromRightSideToUpwardSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the detail mapping.

\* right(i\*3+1) temp(i) upward(i+3) leftward(i\*3+1) downward(5-i) right(i\*3+1)

\* 0 -> 0 -> 6 -> 8 -> 2 -> 0

\* 3 -> 1 -> 7 -> 5 -> 1 -> 3

\* 6 -> 2 -> 8 -> 2 -> 0 -> 6

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=rightwardSide[i\*3];

}

**for**(**int** i=0;i<3;i++){

rightwardSide[i\*3]=downwardSide[2-i];

downwardSide[2-i]=leftwardSide[(2-i)\*3+2];

leftwardSide[(2-i)\*3+2]=upwardSide[6+i];

upwardSide[i+6]=temp[i];

}

/\*\*

\* for forward side,it will be different as well.Here come the details.

\* i= 0 1 2 2 5 8 index=i\*3+2

\* i= 3 4 5 ===>> 1 4 7 index=(i-3)\*3+1=i\*3-8

\* i= 6 7 8 0 3 6 index=(i-6)\*3

\*/

**char**[] tempSide =**new** **char**[9];

**for**(**int** i=0;i<9;i++){

tempSide[i]=forwardSide[i];

}

**for**(**int** i=0;i<9;i++){

**if**(i<3)

forwardSide[i]=tempSide[i\*3+2];

**else** **if**(i<6)

forwardSide[i]=tempSide[(i-3)\*3+1];

**else**

forwardSide[i]=tempSide[(i-6)\*3];

}

System.***out***.println("-->Front layer has Rotated from right side to upward side.");

}

**public** **void** centerLayerRotatesFromRightSideToUpwardSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the detail mapping.

\* right(i\*3+1) temp(i) upward(i+3) leftward(i\*3+1) downward(5-i) right(i\*3+1)

\* 1 -> 0 -> 3 -> 7 -> 5 -> 0

\* 4 -> 1 -> 4 -> 4 -> 4 -> 3

\* 7 -> 2 -> 5 -> 1 -> 3 -> 6

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=rightwardSide[i\*3];

}

**for**(**int** i=0;i<3;i++){

rightwardSide[i\*3+1]=downwardSide[5-i];

downwardSide[5-i]=leftwardSide[i\*3+1];

leftwardSide[i\*3+1]=upwardSide[i+3];

upwardSide[i+3]=temp[i];

}

System.***out***.println("-->Center layer has Rotated from right side to upward side.");

}

**public** **void** backLayerRotatesFromRightSideToUpwardSide() {

**char**[] temp = **new** **char**[3];

/\*\*

\* swap the four side unit's color,and here are the detail mapping.

\* right(i\*3+1) temp(i) upward(i+3) leftward(i\*3+1) downward(5-i) right(i\*3+1)

\* 2 -> 0 -> 0 -> 6 -> 8 -> 2

\* 5 -> 1 -> 1 -> 3 -> 7 -> 5

\* 8 -> 2 -> 2 -> 0 -> 6 -> 8

\*/

**for**(**int** i=0;i<3;i++){

temp[i]=rightwardSide[i\*3];

}

**for**(**int** i=0;i<3;i++){

rightwardSide[i\*3+2]=downwardSide[8-i];

downwardSide[8-i]=leftwardSide[(2-i)\*3];

leftwardSide[(2-i)\*3]=upwardSide[i];

upwardSide[i]=temp[i];

}

/\*\*

\* for backward side,it will be different as well.Here come the details.

\* i= 8 7 6 2 5 8 index=8-(i-6)\*3=26-i\*3

\* i= 5 4 3 ===>> 1 4 7 index=8-((i-3)\*3+1)=16-i\*3

\* i= 2 1 0 0 3 6 index=8-(i\*3+2)=6-i\*3

\*/

**char**[] tempSide =**new** **char**[9];

**for**(**int** i=0;i<9;i++){

tempSide[i]=backwardSide[i];

}

**for**(**int** i=0;i<9;i++){

**if**(i<3)

backwardSide[i]=tempSide[26-i\*3];

**else** **if**(i<6)

backwardSide[i]=tempSide[16-i\*3];

**else**

backwardSide[i]=tempSide[6-i\*3];

}

System.***out***.println("-->Back layer has Rotated from right side to upward side.");

}

/\*\*output on the console as expansion view of cube surface

\* \*/

**public** **void** show(){

// print the title

System.***out***.println("Now here is the rubik's cube \n in view of surface expansion.");

// print the backward side

System.***out***.println(" ---------");

**for**(**int** i=2;i>=0;i--){

System.***out***.println(" "+backwardSide[i\*3+2]+" "+backwardSide[i\*3+1]+" "+backwardSide[i\*3]);

}

System.***out***.println(" ---------");

// print the upward side

**for**(**int** i=0;i<3;i++){

System.***out***.println(" "+upwardSide[i\*3]+" "+upwardSide[i\*3+1]+" "+upwardSide[i\*3+2]);

}

System.***out***.println(" ---------");

// print the left, fowward and right sides

**for**(**int** i=0;i<3;i++){

System.***out***.print(leftwardSide[i\*3]+" "+leftwardSide[i\*3+1]+" "+leftwardSide[i\*3+2]+" | ");

System.***out***.print(forwardSide[i\*3]+" "+forwardSide[i\*3+1]+" "+forwardSide[i\*3+2]+" | ");

System.***out***.println(rightwardSide[i\*3]+" "+rightwardSide[i\*3+1]+" "+rightwardSide[i\*3+2]);

}

System.***out***.println(" ---------");

// print the downward side

**for**(**int** i=0;i<3;i++){

System.***out***.println(" "+downwardSide[i\*3]+" "+downwardSide[i\*3+1]+" "+downwardSide[i\*3+2]);

}

System.***out***.println(" ---------\n");

}

**public** **void** test(){

// print the title

System.***out***.println("Now here is the rubik's cube \n in view of surface expansion.");

// print the backward side

System.***out***.println(" ---------");

**for**(**int** i=2;i>=0;i--){

System.***out***.println(" "+(i\*3+2)+" "+(i\*3+1)+" "+(i\*3));

}

System.***out***.println(" ---------");

// print the upward side

**for**(**int** i=0;i<3;i++){

System.***out***.println(" "+(i\*3)+" "+(i\*3+1)+" "+(i\*3+2));

}

System.***out***.println(" ---------");

// print the left, fowward and right sides

**for**(**int** i=0;i<3;i++){

System.***out***.print((i\*3)+" "+(i\*3+1)+" "+(i\*3+2)+" | ");

System.***out***.print((i\*3)+" "+(i\*3+1)+" "+(i\*3+2)+" | ");

System.***out***.println((i\*3)+" "+(i\*3+1)+" "+(i\*3+2));

}

System.***out***.println(" ---------");

// print the downward side

**for**(**int** i=0;i<3;i++){

System.***out***.println(" "+(i\*3)+" "+(i\*3+1)+" "+(i\*3+2));

}

System.***out***.println(" ---------");

}

**public** **static** **void** main(String[] args){

RublikCube rublikCube=**new** RublikCube(); rublikCube.rightLayerRotatesFromForwardSideToUpwardSide();

rublikCube.show();

rublikCube.topLayerRotatesfromforwardSideToRightSide();

rublikCube.show();

/\*

rublikCube.leftLayerRotatesFromForwardSideToUpwardSide();

rublikCube.show();

rublikCube.bottomLayerRotatesFromForwardSideToRightSide();

rublikCube.show();

\*/

}

}

151224

16121:

componnent of rubic's cube :

1. 中心块： 1 \* 6 =6

2. 角块： 4 \* 2 = 8

3. 棱块： 4 \* 1/2 \* 6 = 12

逻辑块数：6 + 8 + 12 = 26 = 3 \* 3 \* 3 - 1 = 物理构造块数

借助用立体几何里的平面的方向向量思想表示各魔方块：

中心块：

{

(1,0,0),

(0,1,0),

(0,0,1),

(-1,0,0),

(0,-1,0),

(0,0,-1)

}

角块：

{

{(1,0,0),(0,-1,-0),(0,0,1)},

{(1,0,0),(0,1,0),(0,0,1)},

{(1,0,0),(0,1,0),(0,0,-1)},

{(1,0,0),(0,-1,0),(0,0,-1)}

{(-1,0,0),(0,-1,0),(0,0,1)},

{(-1,0,0),(0,1,0),(0,0,1)},

{(-1,0,0),(0,1,0),(0,0,-1)},

{(-1,0,0),(0,-1,0),(0,0,-1)

}

棱块：

{

{(1,0,0),(0,0,1)},

{(1,0,0),(0,1,0)},

{(1,0,0),(0,0,-1)},

{(1,0,0),(0,-1,0)},

{(-1,0,0),(0,0,1)},

{(-1,0,0),(0,1,0)},

{(-1,0,0),(0,0,-1)},

{(-1,0,0),(0,-1,0)}

}

可用统一的3\*3维向量坐标表示：

中心块：

{

{(1,0,0),(0,0,0),(0,0,0)},

{(0,1,0),(0,0,0),(0,0,0)},

{(0,0,1),(0,0,0),(0,0,0)},

{(-1,0,0),(0,0,0),(0,0,0)},

{(0,-1,0),(0,0,0),(0,0,0)},

{(0,0,-1),(0,0,0),(0,0,0)}

}

角块：

{

{(1,0,0),(0,-1,-0),(0,0,1)},

{(1,0,0),(0,1,0),(0,0,1)},

{(1,0,0),(0,1,0),(0,0,-1)},

{(1,0,0),(0,-1,0),(0,0,-1)}

{(-1,0,0),(0,-1,0),(0,0,1)},

{(-1,0,0),(0,1,0),(0,0,1)},

{(-1,0,0),(0,1,0),(0,0,-1)},

{(-1,0,0),(0,-1,0),(0,0,-1)

}

棱块：

{

{(1,0,0),(0,0,1),(0,0,0)},

{(1,0,0),(0,1,0),(0,0,0)},

{(1,0,0),(0,0,-1),(0,0,0)},

{(1,0,0),(0,-1,0),(0,0,0)},

{(-1,0,0),(0,0,1),(0,0,0)},

{(-1,0,0),(0,1,0),(0,0,0)},

{(-1,0,0),(0,0,-1),(0,0,0)},

{(-1,0,0),(0,-1,0),(0,0,0)},

}