卒業研究報告書

題目

テトリスにおける AI の開発

指導教員

石水 隆　講師

報告者

13–1–037–0113

川原 翔太

近畿大学理工学部情報学科

平成 28 年 2 月 3 日提出

概要

テトリスは落ち物ゲームの代表であり、画面上部から降ってくるテトリミノと呼ばれるミノを操作し、適切な場所に積むゲームである。プレイヤーは落下中のミノを左右に移動、または回転させることができる。ミノが画面下部の床か他のミノに触れるとそのミノは固定され、上部より新たなミノが降ってくる。また、ミノを積んだとき、横１列を全てミノで埋めるとその列は消滅する。

テトリスはゲームのレベルに応じて展開されるランダムな局面において、プレイヤーに速い判断力や反射神経などのスキルが要求される。落ちてくるミノもランダムなためゲームをするごとに局面は変化する。

テトリスは脳の刺激にいいとされているが実際にゲームをやり込んでいる人はそう多くは存在しない。

本研究では独自の局面判定を行ない、初心者にむけて配置先の候補を表示し手助けをするプログラムの作成を行なっている。これにより初心者に対してゲームに馴染んでもらう機会を増やしたいと思う。

目次

1 序論

1.1 本研究の背景

テトリスは落ち物ゲームの代表であり、画面上部から降ってくるテトリミノと呼ばれるブロックを操作し、適切な場所に積むゲームである。プレイヤーは落下中のミノを左右に移動、または回転させることができる。ミノが画面下部の床か他のミノに触れるとそのミノは固定され、上部より新たなミノが降ってくる。また、ミノを積んだとき、横 1 列を全てミノで埋めるとその列は消滅する。

1.2 テトリスに関する既知の結果

本節ではテトリスに関する既知の結果について述べる。

テトリスは 1985 年に開発されたコンピュータゲームである。人工知能分野の研究対象ともなっており、テトリスを自動でプレイするプログラムの開発が行われている。

テトリスは単純なゲームであるが、パズルゲームとしてみたときに、完全な解を得るのは難しい。

テトリスは得点を競うゲームであり、高得点を取るにはできるだけ多段消しする必要がある。また、長くゲームを続けるには、積み上がるミノの高さをできるだけ低くしなければならない。しかし、テトリミノが落ちてくる順番が判明しているときに、この得点最大化問題や高さ最小化問題は NP 困難であることが Demaine により示されている [5]。NP 困難とは、問題のサイズの多項式時間で解を得ることが難しいとされてる問題のクラスである。すなわち、n 個のテトリミノが落ちてくるときに、n^c 時間で最も得点が高くなる積み方や、最も高さが低くなる積み方を求めることはできないとされる。

また、ミノのパターンによっては、クリアすることができず必ずゲームオーバーになってしまう場合がある。Burgiel は、Z 字型と S 字型のミノが交互に降ってる場合、70000 個以内に必ずゲームオーバーになること示した [7]。

初心者を対象としたデータから初心者のプレイは積まれたミノの高低差が比較的少ない初期段階では、比較的すきまなくミノを積むような操作が可能であることを示していた。しかし積まれたミノに大きな高低差ができてくると、そのような状況に対処する方法や確認する時間がないために適切な操作ができなくなる。このことから次のようなことがいえる。まず、初心者には積まれたミノの低いところへミノを落とす傾向がみられる。これは、次の操作をしやすくするために積まれたミノの後の局面を比較的平面に保とうとするからと考えられる。一方、熟練者は積まれたミノの低いところへミノを落とそうとはするものの、ある程度の積まれるミノの高低差を許容範囲としている。したがって熟練者のデータを用いた場合には、積まれたミノの局面に比較的高低差のあるものが多く見られている。これは、熟練者が現在の局面のみならず先の事も考慮してゲームを行っていることが考えられる。それに対して初心者は 、現在の局面のことしか考えられず、消すことのできる列から順に消していくことを優先し、ゲームを実行していると考えられる。[6]

1.3 本研究の目的

テトリスは上級者であれば，半永久的にゲームを続けることができる。しかし、テトリスは短時間で適切な場所にミノを置く場所を判断しなければならないため、初心者には難しい。

そこで本研究では、初心者の参考になるように各局面でミノを配置する推奨場所を表示する AI を作成する。

1.4 本報告書の構成

本報告書の構成は以下の通りである. ます第２章てテトリスについて述べる。続く第３章で作成したプログラムについて述べ、４章で結果・考察、５章で結論・今後の課題を述べる。

2 テトリスについて

2.1 テトリスとは

テトリスとはランダムに上から降ってくる７種類のミノを下から敷き詰め隙間なく埋めてゆき、１列隙間なく埋まると消してゆきスコアとして換算するゲームである。

プレイヤーは落下中のミノを回転、または左右に移動することができ、即座に落とすこともできる。左右へは、左右の壁やすでに詰まれた他のミノにぶつかる場合は移動できない。また、回転させると他のミノに重なってしまう場合は回転できない。

ミノを落としたときに、横１列が隙間無く埋まるとその段が消え、それより上にあるミノ全体が消した段の分だけ落下する。消し方によっては複数の段をまとめて消すことができ、同時多くの段を消すほど高いスコアが得られる。とくに、４段まとめ消しは『テトリス』と呼ばれ高得点が入る。また、新たなミノが出現から落とすまでの時間が短い場合もスコアが得られる。従って、テトリスで高得点を得るには、短い時間で多段消しができる場所にミノを落としていくことが必要となる。

2.2 研究内容

本研究では Java を用いてテトリス AI を作成する。本研究で作成するテトリス AI は、画面上部にミノが出現したときに、ミノの配置可能場所を以下の 3 段階に分けて表示する。

1. １段以上消せる場所
2. 隙間なく置ける場所
3. 置かない方が良いと思える場所

上記のより１、２の場所推奨場所であるので, プレイヤーはそこに配置することを目指せばよい。また３を見ることで１、２以外で自分が置いた方がいいと思うところ置ける判断基準を用意した。

現在積まれているミノから局面を判定し、１段以上消せる場合がある場合、その候補場所の色を赤色にして表示する。また、隙間なく置ける場所がある場合、その場所を候補として色を変えて表示する。このときミノの回転角度に応じて候補先の色を変える。画面に表示される隙間無く置ける候補場所の色を表 1 に表示する。一方、現在積まれているミノのうち最も高い場所については、そのミノ付近をあまり置かない方が良いことを示すために灰色で表示する。本研究で作成するテトリスは、プレイヤーはミノを左右に移動、回転、即座に落下の操作を行なうことが出来る。表 2 に本研究で作成するテトリスのキー入力を示す。

表 1　表示される候補場所の色

候補場所 \ 回転角度	0°	90°	180°	270°
隙間無く置ける場所	黄色	緑	ピンク	オレンジ

表 2　キー入力

7	8 90° 回転	9
4 左へ移動	5	6 右へ移動
1	2 落下	3

3　テトリスプログラム

本章では、本研究で作成したテトリスプログラムについて説明する。付録に本研究で作成したテトリスログラムのソースを示す。

プログラムの主要なフィードル変数および各メソッドについて、以下の節で述べる。

3.1　主要なフィールド変数

本節では主要なフィールド変数について、その役割を説明する。

- int　blno = 0 ： ミノの番号
- int　rot = 0 ： 回転角番号 (0=0 度,1=90 度,2=180 度,3=270 度)
- int　block[][] = new int[4][4] ： ミノの配列
- int　bx=3, by=0 ： ミノの位置
- int　nw=24, nh=33 ： ミノの位置の最大数
- int　board[][] = new int[nh+1][nw+1]： 盤面のブロックの有無
- int　square = 8 ： ミノの幅
- int　score ： 点数
- Dimension d ： 表示領域
- Image offs ： オフスクリーン
- Graphics grf
- Thread kicker = null ： アニメーションのためのスレッド変数
- int　speed = 300 ： スピード
- boolean 型 loop = true ： 繰り返すための変数

3.2 init メソッド

init メソッドはまず block[][] の値をすべて 0 にし、画面の中にミノがない状態にするメソッドである。その後、左右の壁、床の壁の値を設定し固定ブロックとする。

さらにオフスクリーンの設定やキーリスナーとして自分自身の登録も行なう。

3.3 blset メソッド

blset メソッドはミノの種類を番号で表す blno とミノの x 位置 bx の値をランダムにセットするメソッドである。

また blset メソッドでは block[][] の中身を初期化する。

3.4 ブロックのセット

blno に応じて block[][] のセットするメソッドを呼び出す。block[][] のセットする中身は blpt0~blpt6 までの７種類のメソッドである。４×４のマス目に対してミノの形に合わせて block[][] の値を 1 にする。回転状態を表す変数 rot によって、ミノの形も変えている。

表 3 に各ミノの形を示す。

表 3　各ミノの形

blpt0	blpt1	blpt2	blpt3	blpt4	blpt5	blpt6
□□□□	□□□□	□□□□	□□□□	■□□□	□□□□	□□□□
□■□□	■□□□	□□□□	□□□□	■□□□	□□□□	□□□□
□■□□	■□□□	□■□□	■■□□	■□□□	□■■□	■■□□
■■□□	■■□□	■■■□	■■□□	■□□□	■■□□	□■■□

3.5 judge メソッド

judge1~4 のメソッドではミノが壁に触れるかどうかの判定を行なう。各メソッドは 1 から順に下側、左側、右側、回転時に触れたかどうかを判定している。

これはミノが 1 つ進んだところに壁があるかを調べている。

3.6 lineblt メソッド

lineblt メソッドは 1 行ごとに全てのマス目が固定ミノ (block[][]＝2) になっているかどうかをチェックしている。

全て固定ミノだった場合、1 行ずつ board[][] の値を下にずらすことによってミノを消す。ずらし終えた上の行はミノのない状態 (board[][]＝0) にする。

3.7 pile メソッド

pile メソッドは現在の局面でミノが置かれた y 座標の 1 個上の y 座標を x 座標ごとに pile[] に保存している。

3.8 hyouka メソッド

hyouka メソッドは pile[] の値を使いミノが置かれた時に隙間が出来るかを判定し隙間が出来ないと判断したら、候補として表示する。ミノの rot の値によって色を変えて表示する。さらにその候補の中で一列以上消せると判断した場合はさらに色を変えてわかりやすく表示する。

3.9 btop メソッド

btop メソッドは現在の局面において積み上げられたミノのなかで 1 番高いミノの y 座標を求める。その y 座標の 1 個上の y 座標の色を変えわかりやすく見せている。

3.10 実行結果

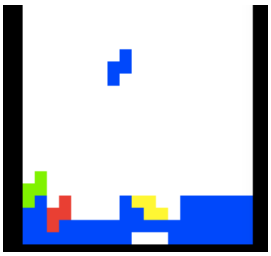

図 1　実行結果 1

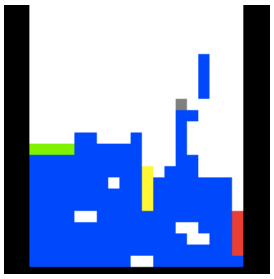

図 2　実行結果 2

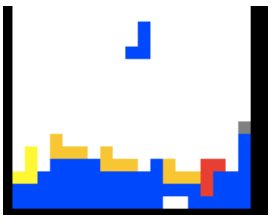

図 3　実行結果 2

以下の図 1,2,3 に実行結果の例を示す。図に示される通り、 1 段以上消せる場所は赤色、隙間なく置ける場所はミノの回転角度によって黄色、オレンジ色、緑色で表示されている。一方、最も高い置くべきでない場所は灰色で表示している。

このように目的の候補場所を複数に分けて表示することに成功した。

4 結果・考察

現在の探索法では pile[] の値を中心に探索を行なっているが、ミノの隣接面でしか隙間なく置けるかの判定しか行なえていない。なので探索外で隙間が表示されたることが起きるときがある。よって明らかにそこに置くとその後のプレイが難しくなる部分のでも候補として表示してしまう。隙間なく置ける部分は全て表示してしまう (ある程度は判定の部分で消しているが) ので rot の値によって候補先の色を変えているがたくさんの候補が出た時や候補同士が重なった時に見にくくなってしまう現象が起きる。

しかし後半になってくると置く場所までの判断時間が短くなるので手助けになる場面が多くなることも確か

だった。

5　結論・今後の課題

今回行なったのはあくまで初心者向けの隙間が生まれないための候補表示であり、それ以降ゲームが長く続くような局面評価は行なえてはいない。現在表示されている候補の中からより良い候補を抽出し表示することで初心者にはすごくやりやすい環境となるだろう。またその候補へ自動で動かすことでテトリスの AI として動かすことも可能だ。

またネクストミノを表示しそれに対しても評価を行い、候補として表示することが理想だろう。

さらにテトリスにはブロックが地面に着いた瞬間固定されるまでほんの少し時間が設けられ、横にスライドすることが出来るのが一般的だが実装しているやり方ではその行動を踏まえた候補表示ができない。従ってこの横スライドに対応させることも今後の課題に挙げられる。

謝辞

本研究を行うにあたって、近畿大学理工学部情報学科情報論理工学研究室の石水 隆講師には大変お世話になりました。研究に関する指摘やサポート、アドバイスなど適切な指導をしていただいたことを、ここに感謝をこめて記します。

参考文献

[1] 村山要司：楽しく学べる Java ゲーム・アプレット，工学社 (2002)

[2] 長久勝：Java ゲームプログラミング，SB クリエイティブ (2007)

[3] 中山亮士，平原誠, 遺伝的アルゴリズムを用いたテトリスの解法, 電子情報通信学会 2015 年総合大会, 情報・システムソサイエティ特別企画 学生ポスターセッション予稿集, p.51, 電子情報通信学会, (2015), https://www.ieice.org/iss/jpn/Publications/issposter_2015/data/pdf/ISS-P-51.pdf

[4] 宮崎真奈実, 荒川正幹, ニューラルネットワークと遺伝的アルゴリズムを用いたテトリスコントローラの開発, 第 74 回全国大会講演論文集, Vol.2012, No.1, pp.539-540, 情報処理学会, (2012), http://id.nii.ac.jp/1001/00109944/

[5] Erik D. Demaine, Susan Hohenberger, David Liben-Nowell, Tetris is Hard, Even to Approximate, Computer Science Vol.2002, No.20 pp,1-56, Cornell Univesity Lbrary, (2002), https://arxiv.org/abs/cs/0210020

[6] 田伏未来, 萩原将文, ファジィ推論ニューラルネットワークを用いたテトリスのスキル獲得のための自動学習, 日本ファジィ学会誌 Vol.11, No,6, pp.1089-1097, 日本ファジィ学会, (1999), http://ci.nii.ac.jp/naid/110002946575

[7] Heidi Burgiel, How to lose at Tetris, The Mathematical Gazette, Vol.81, No.491, pp.194-200, (1997), http://www.geom.uiuc.edu/java/tetris/tetris.ps

[8] 遠城秀和, 実時間知識処理をめざした制約推論のレスポンスタイム推定法, 全国大会講演論文集, 第 44 回 (人工知能及び認知科学), pp.7-8, 情報処理学会, (1992), http://id.nii.ac.jp/1001/00121333/

付録 A　付録

本研究で作成したテトリスプログラムのソースプログラムを以下に示す。

```
import java.applet.Applet;
import java.awt.*;
import java.util.*;
import java.awt.event.*;

public class otoshi2 extends java.applet.Applet implements Runnable,
 KeyListener, MouseListener{

        int blno=0; // ブロック No.
        int blnok=0; //仮ブロック No.
        int rot=0; // 回転角番号 (0=0 度,1=90 度,2=180 度,3=270 度)
        int block[][] = new int[4][4]; // ブロック
        int blockk[][] = new int[4][4]; //仮ブロック
        int bx=3, by=0; // ブロックの位置
        int kouhox,kouhoy;
        int top[] = new int[1];
        int btopy;
        int btopx;
        int pile[] = new int[19];
        int yy;
        int hyouka[][] = new int[5][5];

        int counter;
        int counter2;
        int counter3;
        int counter4;

        int nw=24, nh=33; // ブロック位置の最大数
        int board[][] = new int[nh+1][nw+1]; // 盤面のブロックの有無
        int square = 8; // ブロックの幅

        int score; // 点数

        Dimension d;  // 表示領域
        Image  offs;  // オフスクリーン
        Graphics grf;

        Thread kicker = null; // アニメーションのためのスレッド変数
        int speed = 300; // スピード
        boolean loop = true;      // 繰り返すための変数



        public void init() {
                int i ,j; // カウンター
```

```
        /* ブロックがあるかないかの設定 */
        for( i = 0 ; i <= nh ; i++ ){
                for( j = 0 ; j <= nw ; j++ ){
                        board[i][j] = 0; // ブロックがない
                }
        }

        /* 左右、下の固定ブロック */
        for( i = 0 ; i <= nh ; i++ ){
                for( j = 0 ; j < 3 ; j++ ){
                        board[i][j] = 2; // 左固定ブロック
                        board[i][nw-j] = 2; // 右固定ブロック
                }
        }
        for( i = 0 ; i < 3 ; i++ ){
                for( j = 0 ; j <= nw ; j++ ){
                        board[nh-i][j] = 2; // 下固定ブロック
                }
        }



        /* オフスクリーンの設定 */
        d = getSize(); // 表示領域の取得
        offs = createImage( d.width, d.height);
        grf  = offs.getGraphics();

        /* キーリスナとして自分自身を登録 */
        addKeyListener(this);
        requestFocus(); // キー入力フォーカス

        /* マウスリスナとして自分自身を登録 */
        addMouseListener(this);

        /* ブロックの設定 */
        blset();

        score=0;

        loop = true;
        start();

}

public void paint(Graphics g) {
        update(g);
}

public void update(Graphics g) {
        int i, j; // カウンター

        /* バックをオレンジで塗る */
        grf.setColor(Color.white);
```

```
grf.fillRect(0,0,d.width, d.height);

/* ブロックを描く */

grf.setColor(Color.blue);
for( i = 0 ; i <= nh ; i++ ){
        for( j = 0 ; j <= nw ; j++ ){
                if ( board[i][j] >= 1 && board[i][j] <= 2) { // ブ
ロックがある
                        grf.fillRect(j*square, i*square, square, square);
                }
        }
}

grf.setColor(Color.yellow);
for( i = 0 ; i <= nh ; i++ ){
        for( j = 0 ; j <= nw ; j++ ){
                if ( board[i][j] == 4 ) { // 仮ブロックがある
                        grf.fillRect(j*square, i*square, square, square);
                }
        }
}

grf.setColor(Color.green);
for( i = 0 ; i <= nh ; i++ ){
        for( j = 0 ; j <= nw ; j++ ){
                if ( board[i][j] == 5 ) { // 仮ブロックがある
                        grf.fillRect(j*square, i*square, square, square);
                }
        }
}

grf.setColor(Color.pink);
for( i = 0 ; i <= nh ; i++ ){
        for( j = 0 ; j <= nw ; j++ ){
                if ( board[i][j] == 6 ) { // 仮ブロックがある
                        grf.fillRect(j*square, i*square, square, square);
                }
        }
}

grf.setColor(Color.orange);
for( i = 0 ; i <= nh ; i++ ){
        for( j = 0 ; j <= nw ; j++ ){
                if ( board[i][j] == 7 ) { // 仮ブロックがある
                        grf.fillRect(j*square, i*square, square, square);
                }
        }
}


grf.setColor(Color.gray);
for( i = 0 ; i <= nh ; i++ ){
```

```
        for( j = 0 ; j <= nw ; j++ ){
                if ( board[i][j] == 8 ) { // 仮ブロックがある
                        grf.fillRect(j*square, i*square, square, square);
                }
        }
}
grf.setColor(Color.red);
for( i = 0 ; i <= nh ; i++ ){
        for( j = 0 ; j <= nw ; j++ ){
                if ( board[i][j] == 9 ) { // 仮ブロックがある
                        grf.fillRect(j*square, i*square, square, square);
                }
        }
}


/* 左右、下の固定ブロック */
grf.setColor(Color.black);
for( i = 0 ; i <= nh ; i++ ){
        for( j = 0 ; j < 3 ; j++ ){
                /* 左固定ブロック */
                grf.fillRect(j*square, i*square, square, square);
                /* 右固定ブロック */
                grf.fillRect((nw-j)*square, i*square, square, square);
        }
}
for( i = 0 ; i < 3 ; i++ ){
        for( j = 0 ; j <= nw ; j++ ){
                /* 下固定ブロック */
                grf.fillRect(j*square, (nh-i)*square, square, square);
        }
}


/* 点数の表示 */
grf.setColor(Color.white);
grf.drawString("Score : "+score, 120, 265);

if ( !loop ) {
        grf.setColor(Color.white);
        grf.drawString("Game Over", 12, 265);
        /* RESTART ボタンの描画 */
        grf.setColor(Color.red);
        grf.fillRect( 48, 120, 65, 25 );
        grf.setColor(Color.white);
        grf.drawString("RESTART", 55, 138);
        grf.setColor(Color.white);
        grf.drawLine( 48, 120, 113, 120);
        grf.drawLine( 48, 120, 48, 145);
        grf.setColor(Color.black);
        grf.drawLine( 48, 145, 113, 145);
        grf.drawLine( 113, 145, 113, 120);
}
```

```
        /* オフスクリーンのイメージを一挙に実際の表示領域に描く */
        g.drawImage(offs, 0, 0, this);




}

public void keyPressed(KeyEvent e) {
        /* キーが押された */

        int key = e.getKeyCode(); // 押されたキーのコードを取得

        int numkey; // 数字キー
        numkey = key - '0'; // 数字キーの計算

        if ( numkey == 4 ) {
                if ( judge2() ) {
                        bx = bx - 1;
                }
        }

        if ( numkey == 6 ) {
                if ( judge3() ) {
                        bx = bx + 1;
                }
        }

        if ( numkey == 2 ) {
                if ( judge1() ) {
                        by = by +1;
                }
        }

        if ( numkey == 8 ) {
                rot = rot +1;
                if ( rot >= 4 ) {
                        rot = 0;
                }
                /* ブロックの中身をセット */
                switch(blno){
                case 0:
                        blpt0();
                        break;
                case 1:
                        blpt1();
                        break;
                case 2:
                        blpt2();
                        break;
                case 3:
                        blpt3();
```

```
                        break;
                case 4:
                        blpt4();
                        break;
                case 5:
                        blpt5();
                        break;
                case 6:
                        blpt6();
                        break;
                }
                if ( !judge4() ) {
                        rot = rot-1;
                        if ( rot < 0 ) {
                                rot = 4;
                        }
                }
        }
 }

 public void keyTyped(KeyEvent e) {
        /* キーが押されてから離された */
 }

 public void keyReleased(KeyEvent e) {
        /* キーが離された */
 }

public void start() {
        if(kicker == null) {
                /* スレッドを実行させる */
                kicker = new Thread(this);
                kicker.start();


       }
 }

 public void stop() {
        /* スレッドを止める */
        kicker = null;
 }

 public void run() {
        int i ,j; // カウンター
        int line=0; // 消した行数
        int ibx, iby; // 配列のインデックス
        //int ikouhox,ikouhoy;

        /* 実行中のスレッドをチェック */
        Thread thisThread = Thread.currentThread();

        /* ずっと繰り返し */
```

```
while( loop && kicker == thisThread) {

        /* 移動前のブロックを消す */
        for( i = 0 ; i <= nh-3 ; i++ ){
                for( j = 3 ; j <= nw-3 ; j++ ){
                        if ( board[i][j] < 2 /*|| board[i][j] >= 4*/) {
                                board[i][j] = 0; // ブロックがない
                        }
                }
        }




        /* ブロックの中身をセット */
        switch(blno){
                case 0:
                        blpt0();
                        break;
                case 1:
                        blpt1();
                        break;
                case 2:
                        blpt2();
                        break;
                case 3:
                        blpt3();
                        break;
                case 4:
                        blpt4();
                        break;
                case 5:
                        blpt5();
                        break;
                case 6:
                        blpt6();
                        break;
        }




        /* ブロックを進める */
        if ( judge1() ) {
                by = by + 1;
        } else {
                /* 下まで来たら、 */
                /* ブロックを固定 */
                for( i = 0 ; i < 4 ; i++ ){
                        for( j = 0 ; j < 4 ; j++ ){
                                if ( block[i][j] == 1 ) {
                                        ibx = bx+j;
```

```
                                        iby = by+i;
                                        if ( iby >= 0 && ibx >= 0 ) {
                                                board[by+i][bx+j] = 2;
                                                kouhosakujo();
                                        }
                                }
                        }
                }
                // トップ座標の計算
                btop();

                /* 次のブロック */
                blset();

// 候補ブロックの設置
                hyouka();




        }

        /* 移動後のブロックを設定 */
        for( j = 0 ; j < 4 ; j++ ){
                for( i = 0 ; i < 4 ; i++ ){
                        iby = by+i;
                        ibx = bx+j;
                        if ( iby >= 0 && ibx >= 0 ) {
                                if ( board[by+i][bx+j] < 2 ) {
                                        board[by+i][bx+j] = block[i][j];
                                }
                        }
                }
        }


        /* 一行揃っていたら消して詰める */
        for( i = 1 ; i <= nh-3 ; i++ ){
                if ( lineblt(i) ){
                        line=line+1; // 消した行数をカウント
                }
        }
        /* 消した行数から点数を計算 */
        line=line*line;
        score=score+line*10;
        line=0;
        repaint();

        /* 一行めに固定ブロックがあったらゲーム終了 */
        for( j = 3 ; j <= nw-3 ; j++ ){
                if ( board[0][j] == 2 ) {
                        loop = false;
                }
```

```
                }

                try{
                        Thread.sleep(speed);
                } catch (InterruptedException e){}
        }
}

public void blset() {
        int i, j; // カウンター
        blno =(int)( Math.random() * (float)(7) );
        rot = 0;
        bx = 12;
        by = -4;
        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        block[i][j] = 0; // ブロックがない
                }
        }
}


public boolean judge1() {
        int i, j; // カウンター
        int ibx, iby; // 配列のインデックス

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        iby = by+i;
                        ibx = bx+j;
                        if ( iby >= 0 && ibx >= 0 ) {
                                if ( 3 == board[by+1+i][bx+j] + block[i][j] ) {
                                        return false;
                                }
                        }
                }
        }
        return true;
}
public boolean judge2() {
        int i, j; // カウンター
        int ibx, iby; // 配列のインデックス

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        iby = by+i;
                        ibx = bx+j;
                        if ( iby >= 0 && ibx >= 0 ) {
                                if ( 3 == board[by+i][bx-1+j] + block[i][j] ) {
                                        return false;
                                }
                        }
                }
```

```
        }
        return true;
    }
    public boolean judge3() {
        int i, j; // カウンター
        int ibx, iby; // 配列のインデックス
        for( i = 0 ; i < 4 ; i++ ){
            for( j = 0 ; j < 4 ; j++ ){
                iby = by+i;
                ibx = bx+j;
                if ( iby >= 0 && ibx >= 0 ) {
                    if ( 3 == board[by+i][bx+1+j] + block[i][j] ) {
                        return false;
                    }
                }
            }
        }
        return true;
    }
    public boolean judge4() {
        int i, j; // カウンター
        int ibx, iby; // 配列のインデックス

        for( i = 0 ; i < 4 ; i++ ){
            for( j = 0 ; j < 4 ; j++ ){
                iby = by+i;
                ibx = bx+j;
                if ( iby >= 0 && ibx >= 0 ) {
                    if ( 3 == board[by+i][bx+j] + block[i][j] ) {
                        return false;
                    }
                }
            }
        }
        return true;
    }
    public boolean lineblt(int i) {
        int j, k; // カウンター
        for( j = 3 ; j <= nw-3 ; j++ ){
            if ( board[i][j] != 2 ) {
                return false; // 揃っていなかったらこのメソッドを抜け
る
            }
        }
        for( k = 0 ; k < i ; k++ ){
            for( j = 3 ; j <= nw-3 ; j++ ){
                board[i-k][j]=board[i-k-1][j]; // ブロックを詰める
            }
        }
        for( j = 3 ; j <= nw-3 ; j++ ){
            board[0][j] = 0; // 1 行目のセット（ブロックはない）
        }
        return true;
```

```
}

public void mousePressed(MouseEvent e){
        int ix, iy; // マウスが押された座標

        /* マウスが押された座標を得る */
        ix = e.getX();
        iy = e.getY();

        if ( ix > 48 && ix < 113 && iy > 120 && iy < 145 ) {
                /* マウスの座標 (ix,iy) が RESTART ボタン内 */
                /* だったらゲーム再スタート              */
                if (!loop) {
                        stop();
                        init();
                        repaint();
                }
        }
}

public void mouseReleased(MouseEvent e){
        /* マウスボタンが離された */
}

public void mouseClicked(MouseEvent e){
        /* マウスボタンがクリックされた */
}

public void mouseEntered(MouseEvent e){
        /* マウスカーソルが入ってきた */
}

public void mouseExited(MouseEvent e){
        /* マウスカーソルが出ていった */
}

public void blpt0() {
        int i ,j; // カウンター

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        block[i][j] = 0;
                }
        }
        switch(rot){
                case 0:
                        /* blno=0, rot=0 */
                        block[1][1]=1;
                        block[2][1]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        break;
                case 1:
```

```
                        /* blno=0, rot=1 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[2][2]=1;
                        block[3][2]=1;
                        break;
                case 2:
                        /* blno=0, rot=2 */
                        block[1][0]=1;
                        block[1][1]=1;
                        block[2][0]=1;
                        block[3][0]=1;
                        break;
                case 3:
                        /* blno=0, rot=3 */
                        block[2][0]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        block[3][2]=1;
                        break;
        }
}
public void blpt1() {
        int i ,j; // カウンター

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        block[i][j] = 0;
                }
        }
        switch(rot){
                case 0:
                        /* blno=1, rot=0 */
                        block[1][0]=1;
                        block[2][0]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        break;
                case 1:
                        /* blno=1, rot=1 */
                        block[2][2]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        block[3][2]=1;
                        break;
                case 2:
                        /* blno=1, rot=2 */
                        block[1][0]=1;
                        block[1][1]=1;
                        block[2][1]=1;
                        block[3][1]=1;
                        break;
                case 3:
```

```
                        /* blno=1, rot=3 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[2][2]=1;
                        block[3][0]=1;
                        break;
        }
}
public void blpt2() {
        int i ,j; // カウンター

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        block[i][j] = 0;
                }
        }
        switch(rot){
                case 0:
                        /* blno=2, rot=0 */
                        block[2][1]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        block[3][2]=1;
                        break;
                case 1:
                        /* blno=2, rot=1 */
                        block[1][1]=1;
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][1]=1;
                        break;
                case 2:
                        /* blno=2, rot=2 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[2][2]=1;
                        block[3][1]=1;
                        break;
                case 3:
                        /* blno=2, rot=3 */
                        block[1][0]=1;
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][0]=1;
                        break;
        }
}
public void blpt3() {
        int i ,j; // カウンター

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        block[i][j] = 0;
```

```
                }
        }
        switch(rot){
                case 0:
                        /* blno=3, rot=0 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        break;
                case 1:
                        /* blno=3, rot=1 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        break;
                case 2:
                        /* blno=3, rot=2 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        break;
                case 3:
                        /* blno=3, rot=3 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        break;
        }
}
public void blpt4() {
        int i ,j; // カウンター

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        block[i][j] = 0;
                }
        }
        switch(rot){
                case 0:
                        /* blno=4, rot=0 */
                        block[0][0]=1;
                        block[1][0]=1;
                        block[2][0]=1;
                        block[3][0]=1;
                        break;
                case 1:
                        /* blno=4, rot=1 */
                        block[3][0]=1;
                        block[3][1]=1;
```

```
                        block[3][2]=1;
                        block[3][3]=1;
                        break;
                case 2:
                        /* blno=4, rot=2 */
                        block[0][0]=1;
                        block[1][0]=1;
                        block[2][0]=1;
                        block[3][0]=1;
                        break;
                case 3:
                        /* blno=4, rot=3 */
                        block[3][0]=1;
                        block[3][1]=1;
                        block[3][2]=1;
                        block[3][3]=1;
                        break;
        }
}
public void blpt5() {
        int i ,j; // カウンター

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        block[i][j] = 0;
                }
        }
        switch(rot){
                case 0:
                        /* blno=5, rot=0 */
                        block[2][1]=1;
                        block[2][2]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        break;
                case 1:
                        /* blno=5, rot=1 */
                        block[1][0]=1;
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][1]=1;
                        break;
                case 2:
                        /* blno=5, rot=2 */
                        block[2][1]=1;
                        block[2][2]=1;
                        block[3][0]=1;
                        block[3][1]=1;
                        break;
                case 3:
                        /* blno=5, rot=3 */
                        block[1][0]=1;
                        block[2][0]=1;
```

```
                        block[2][1]=1;
                        block[3][1]=1;
                        break;
        }
}
public void blpt6() {
        int i ,j; // カウンター

        for( i = 0 ; i < 4 ; i++ ){
                for( j = 0 ; j < 4 ; j++ ){
                        block[i][j] = 0;
                }
        }
        switch(rot){
                case 0:
                        /* blno=6, rot=0 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][1]=1;
                        block[3][2]=1;
                        break;
                case 1:
                        /* blno=6, rot=1 */
                        block[1][1]=1;
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][0]=1;
                        break;
                case 2:
                        /* blno=6, rot=2 */
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][1]=1;
                        block[3][2]=1;
                        break;
                case 3:
                        /* blno=6, rot=3 */
                        block[1][1]=1;
                        block[2][0]=1;
                        block[2][1]=1;
                        block[3][0]=1;
                        break;
        }
}
public void kouhosakujo(){
        int i, j;
        for( i = 0 ; i <= nh ; i++ ){
                for( j = 0 ; j <= nw ; j++ ){
                        if ( board[i][j] >= 4) { // ブロックがある
                                board[i][j] = 0;
                        }
                }
        }
```

```
	}
	public int btop(){

		boolean find = false;
		int i, j;
		btopy = 0;
		btopx = 0;
		out:{
		for(i = 0; i <= nh-3; i++){
			for(j = 3; j <= nw-3; j++){
				if(board[i][j]==2){
				btopy = i;
				btopx = j;
				find = true;
			}
				if(find)break out;
			}
		}
		}
		board[btopy-1][btopx] = 8;

		return btopy;
	}

	public void hyouka(){
		int bx;
		int i,j,k;

		for(bx = 3; bx <= nw - 3; bx++){
			pile[bx-3] = pile(bx);
			}
		for(i = 3; i <= nw -3; i++){
			for(k = 0; k < 5; k++){
				for(j = 0; j < 5; j++){
				hyouka[k][j] = 0;
			}
			}
			if(pile[i-3] != 30){
			if(board[pile[i-3]-2][i-2]  == 0 || board[pile[i-3]-2][i-2]  == 8)
hyouka[0][0] = 1;
			if(board[pile[i-3]-2][i-1]  == 0 || board[pile[i-3]-2][i-1]  == 8)
hyouka[0][1] = 1;
			if(board[pile[i-3]-2][i] == 0 || board[pile[i-3]-2][i] == 8)
hyouka[0][2] = 1;
			if(board[pile[i-3]-2][i+1] == 0 || board[pile[i-3]-2][i+1] == 8)
hyouka[0][3] = 1;
			if(board[pile[i-3]-2][i+2] == 0 || board[pile[i-3]-2][i+2] == 8)
hyouka[0][4] = 1;
			if(board[pile[i-3]-1][i-2]  == 0 || board[pile[i-3]-1][i-2]  == 8)
hyouka[1][0] = 1;
			if(board[pile[i-3]-1][i-1]  == 0 || board[pile[i-3]-1][i-1]  == 8)
hyouka[1][1] = 1;
			if(board[pile[i-3]-1][i] == 0 || board[pile[i-3]-1][i] == 8)
```

```
 hyouka[1][2] = 1;
                        if(board[pile[i-3]-1][i+1] == 0 || board[pile[i-3]-1][i+1] == 8)
 hyouka[1][3] = 1;
                        if(board[pile[i-3]-1][i+2]  == 0 || board[pile[i-3]-1][i+2]  == 8)
 hyouka[1][4] = 1;
                        if(board[pile[i-3]][i-2] == 0 || board[pile[i-3]][i-2] == 8)
 hyouka[2][0] =1;
                        if(board[pile[i-3]][i-1] == 0 || board[pile[i-3]][i-1] == 8)
 hyouka[2][1] =1;
                        if(board[pile[i-3]][i] == 0 || board[pile[i-3]][i] == 8)
 hyouka[2][2] = 1;
                        if(board[pile[i-3]][i+1] == 0 || board[pile[i-3]][i+1] == 8)
 hyouka[2][3] = 1;
                        if(board[pile[i-3]][i+2] == 0 || board[pile[i-3]][i+2] == 8)
 hyouka[2][4] = 1;
                        if(board[pile[i-3]+1][i-2] == 0 || board[pile[i-3]+1][i-2] == 8)
 hyouka[3][0] = 1;
                        if(board[pile[i-3]+1][i-1] == 0 || board[pile[i-3]+1][i-1] == 8)
 hyouka[3][1] = 1;
                        if(board[pile[i-3]+1][i]  == 0 || board[pile[i-3]+1][i]  == 8)
 hyouka[3][2] = 1;
                        if(board[pile[i-3]+1][i+1] == 0 || board[pile[i-3]+1][i+1] == 8)
 hyouka[3][3] = 1;
                        if(board[pile[i-3]+1][i+2] == 0 || board[pile[i-3]+1][i+2] == 8)
 hyouka[3][4] = 1;
                        if(board[pile[i-3]+2][i-2] == 0 || board[pile[i-3]+2][i-2] == 8)
 hyouka[4][0] = 1;
                        if(board[pile[i-3]+2][i-1] == 0 || board[pile[i-3]+2][i-1] == 8)
 hyouka[4][1] = 1;
                        if(board[pile[i-3]+2][i]  == 0 || board[pile[i-3]+2][i]  == 8)
 hyouka[4][2] = 1;
                        if(board[pile[i-3]+2][i+1] == 0 || board[pile[i-3]+2][i+1] == 8)
 hyouka[4][3] = 1;
                        if(board[pile[i-3]+2][i+2] == 0 || board[pile[i-3]+2][i+2] == 8)
 hyouka[4][4] = 1;
                        }
                switch(blno){
                case 0:
                        if(hyouka[2][2] == 1 && hyouka[2][1] == 1 && hyouka[1][2] == 1 &&
hyouka[0][2] == 1 && board[pile[i-3]+1][i-1] == 2 && board[pile[i-3]+1][i] == 2
&& board[pile[i-3]][i+1] == 2){
                                board[pile[i-3]][i] = 4;
                                board[pile[i-3]][i-1] = 4;
                                board[pile[i-3]-1][i] = 4;
                                board[pile[i-3]-2][i] = 4;

                                counter = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i-1] = 9;
```

```
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                }
                        }
                        if( hyouka[2][2] == 1 && hyouka[2][3] == 1 && hyouka[2][4] == 1 &&
hyouka[3][4] == 1 && board[pile[i-3]+1][i] == 2 && board[pile[i-3]+1][i+1] == 2){
                                board[pile[i-3]][i] = 5;
                                board[pile[i-3]][i+1] = 5;
                                board[pile[i-3]][i+2] = 5;
                                board[pile[i-3]+1][i+2] = 5;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 16){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]][i+2] = 9;
                                        board[pile[i-3]+1][i+2] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]+1][m] == 2)counter2++;
                                }
                                if(counter2 == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]][i+2] = 9;
                                        board[pile[i-3]+1][i+2] = 9;
                                }
                        }
                        if( hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[0][2] == 1 &&
hyouka[0][3] == 1 && board[pile[i-3]-1][i+1] == 2 && board[pile[i-3]][i+1] == 2){
                                board[pile[i-3]][i] = 6;
                                board[pile[i-3]-1][i] = 6;
                                board[pile[i-3]-2][i] = 6;
                                board[pile[i-3]-2][i+1] = 6;
                                counter = 0;
                                counter2 = 0;
                                counter3 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-2][i+1] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 18){
```

```
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-2][i+1] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-2][m] == 2)counter3++;
                                }
                                if(counter3 == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-2][i+1] = 9;
                                }
                                }
                        if( hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[2][3] == 1
&& hyouka[2][4] == 1 && board[pile[i-3]+1][i] == 2 && board[pile[i-3]+1][i+1] == 2 &&
board[pile[i-3]+1][i+2] == 2 ){
                                board[pile[i-3]][i] = 7;
                                board[pile[i-3]-1][i] = 7;
                                board[pile[i-3]][i+1] = 7;
                                board[pile[i-3]][i+2] = 7;
                                counter = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 16){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]][i+2] = 9;
                                }
                        }
                        break;
                case 1:
                        if(hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[0][2] == 1 &&
hyouka[2][3] == 1 && board[pile[i-3]+1][i] == 2 && board[pile[i-3]+1][i+1] == 2 &&
board[pile[i-3]][i-1] == 2){
                                board[pile[i-3]][i] = 4;
                                board[pile[i-3]-1][i] = 4;
                                board[pile[i-3]-2][i] = 4;
                                board[pile[i-3]][i+1] = 4;
                                counter = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                }
                        }
```

```
                        if( hyouka[2][2] == 1 && hyouka[2][3] == 1 && hyouka[2][4] == 1 &&
hyouka[1][3] == 1 && board[pile[i-3]+1][i] == 2 && board[pile[i-3]+1][i+1] == 2 &&
board[pile[i-3]+1][i+2] == 2){
                                board[pile[i-3]][i] = 5;
                                board[pile[i-3]][i+1] = 5;
                                board[pile[i-3]][i+2] = 5;
                                board[pile[i-3]-1][i+2] = 5;
                                counter = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 16){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]][i+2] = 9;
                                        board[pile[i-3]-1][i+2] = 9;
                                }
                        }
                        if( hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[0][2] == 1 &&
hyouka[0][1] == 1 && board[pile[i-3]-1][i-1] == 2 && board[pile[i-3]][i-1] == 2){
                                board[pile[i-3]][i] = 6;
                                board[pile[i-3]-1][i] = 6;
                                board[pile[i-3]-2][i] = 6;
                                board[pile[i-3]-2][i-1] = 6;
                                counter = 0;
                                counter2 = 0;
                                counter3 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-2][i-1] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-2][i-1] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-2][m] == 2)counter3++;
                                }
                                if(counter3 == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-2][i-1] = 9;
```

```
                                }
                                }
                        if( hyouka[2][2] == 1 && hyouka[2][1] == 1 && hyouka[2][0] == 1 &&
hyouka[3][0] == 1 && board[pile[i-3]+1][i-1] == 2 && board[pile[i-3]+1][i] == 2){
                                board[pile[i-3]][i] = 7;
                                board[pile[i-3]][i-1] = 7;
                                board[pile[i-3]][i-2] = 7;
                                board[pile[i-3]+1][i-2] = 7;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 16){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i-1] = 9;
                                        board[pile[i-3]][i-2] = 9;
                                        board[pile[i-3]+1][i-2] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]+1][m] == 2)counter2++;
                                }
                                if(counter2 == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i-1] = 9;
                                        board[pile[i-3]][i-2] = 9;
                                        board[pile[i-3]+1][i-2] = 9;
                                }
                        }
                        break;
                case 2:
                        if(hyouka[2][2] == 1 && hyouka[2][1] == 1 && hyouka[2][3] == 1 &&
hyouka[1][1] == 1 && board[pile[i-3]+1][i-1] == 2 && board[pile[i-3]+1][i] == 2 &&
board[pile[i-3]+1][i+1] == 2){
                                board[pile[i-3]][i] = 4;
                                board[pile[i-3]][i-1] = 4;
                                board[pile[i-3]][i+1] = 4;
                                board[pile[i-3]-1][i] = 4;
                                counter = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 16){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i-1] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                }
                        }
                        if( hyouka[2][2] == 1 && hyouka[1][1] == 1 && hyouka[1][2] == 1 &&
hyouka[0][2] == 1 && board[pile[i-3]][i-1] == 2 && board[pile[i-3]+1][i] == 2){
                                board[pile[i-3]][i] = 5;
                                board[pile[i-3]-1][i-1] = 5;
```

```
                                board[pile[i-3]-1][i] = 5;
                                board[pile[i-3]-2][i] = 5;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i-1] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i-1] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                }
                        }
                        if( hyouka[2][2] == 1 && hyouka[1][1] == 1 && hyouka[1][2] == 1 &&
hyouka[1][3] == 1 && board[pile[i-3]][i-1] == 2 && board[pile[i-3]+1][i] == 2 &&
board[pile[i-3]][i+1] == 2){
                                board[pile[i-3]][i] = 6;
                                board[pile[i-3]-1][i] = 6;
                                board[pile[i-3]-2][i] = 6;
                                board[pile[i-3]-2][i-1] = 6;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-2][i-1] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 16){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-2][i-1] = 9;
                                }
                                }
                        if( hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[1][3] == 1 &&
hyouka[0][2] == 1 && board[pile[i-3]+1][i] == 2 && board[pile[i-3]][i+1] == 2){
```

```
                                board[pile[i-3]][i] = 7;
                                board[pile[i-3]-1][i] = 7;
                                board[pile[i-3]-1][i+1] = 7;
                                board[pile[i-3]-2][i] = 7;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-1][i+1] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-1][i+1] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                }
                        }
                        break;
                case 3:
                        if(hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[1][1] == 1 &&
hyouka[2][1] == 1 && board[pile[i-3]][i-2] == 2 && board[pile[i-3]+1][i-1] == 2 &&
board[pile[i-3]+1][i] == 2){
                                board[pile[i-3]][i] = 4;
                                board[pile[i-3]-1][i] = 4;
                                board[pile[i-3]-1][i-1] = 4;
                                board[pile[i-3]][i-1] = 4;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-1][i-1] = 9;
                                        board[pile[i-3]][i-1] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-1][i-1] = 9;
                                        board[pile[i-3]][i-1] = 9;
```

```
                                }
                        }
                        if( hyouka[2][2] == 1 && hyouka[2][3] == 1 && hyouka[1][3] == 1 &&
hyouka[1][2] == 1 && board[pile[i-3]+1][i] == 2 && board[pile[i-3]+1][i+1] == 2 &&
board[pile[i-3]][i+2] == 2){
                                board[pile[i-3]][i] = 5;
                                board[pile[i-3]][i+1] = 5;
                                board[pile[i-3]-1][i+1] = 5;
                                board[pile[i-3]-1][i] = 5;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]-1][i+1] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]-1][i+1] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                }
                        }
                        break;
                case 4:
                        if(hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[0][2] == 1 &&
board[pile[i-3]-3][i] == 0 && board[pile[i-3]][i-1] == 2 && board[pile[i-3]+1][i] == 2 &&
board[pile[i-3]][i+1] == 2){
                                board[pile[i-3]][i] = 4;
                                board[pile[i-3]-1][i] = 4;
                                board[pile[i-3]-2][i] = 4;
                                board[pile[i-3]-3][i] = 4;
                                counter = 0;
                                counter2 = 0;
                                counter3 = 0;
                                counter4 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-3][i] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
```

```
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-3][i] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-2][m] == 2)counter3++;
                                }
                                if(counter3 == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-3][i] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-3][m] == 2)counter4++;
                                }
                                if(counter4 == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-2][i] = 9;
                                        board[pile[i-3]-3][i] = 9;
                                }

                        }
                        if( hyouka[2][2] == 1 && hyouka[2][3] == 1 && hyouka[2][4] == 1 &&
board[pile[i-3]][i+3] == 0 && board[pile[i-3]+1][i] == 2 && board[pile[i-3]+1][i+1] == 2 &&
board[pile[i-3]+1][i+2] == 2 && board[pile[i-3]+1][i+3] == 2){
                                board[pile[i-3]][i] = 5;
                                board[pile[i-3]][i+1] = 5;
                                board[pile[i-3]][i+2] = 5;
                                board[pile[i-3]][i+3] = 5;
                                counter = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 15){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]][i+2] = 9;
                                        board[pile[i-3]][i+3] = 9;
                                }
                        }
                        break;
                case 5:
                        if(hyouka[2][2] == 1 && hyouka[2][3] == 1 && hyouka[1][3] == 1 &&
hyouka[1][4] == 1 && board[pile[i-3]][i+2] == 2 && board[pile[i-3]+1][i] == 2 &&
board[pile[i-3]+1][i+1] == 2){
                                board[pile[i-3]][i] = 4;
                                board[pile[i-3]][i+1] = 4;
```

```
                                board[pile[i-3]-1][i+1] = 4;
                                board[pile[i-3]-1][i+2] = 4;
                                counter = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i+1] = 9;
                                        board[pile[i-3]-1][i+1] = 9;
                                        board[pile[i-3]-1][i+2] = 9;
                                }
                        }
                        if( hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[1][1] == 1 &&
hyouka[0][1] == 1 && board[pile[i-3]][i-1] == 2 && board[pile[i-3]+1][i] == 2){
                                board[pile[i-3]][i] = 5;
                                board[pile[i-3]-1][i] = 5;
                                board[pile[i-3]-1][i-1] = 5;
                                board[pile[i-3]-2][i-1] = 5;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-1][i-1] = 9;
                                        board[pile[i-3]-2][i-1] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-1][i-1] = 9;
                                        board[pile[i-3]-2][i-1] = 9;
                                }
                        }
                        break;
                case 6:
                        if(hyouka[2][2] == 1 && hyouka[2][1] == 1 && hyouka[1][1] == 1 &&
hyouka[1][0] == 1 && board[pile[i-3]][i-2] == 2 && board[pile[i-3]+1][i-1] == 2 &&
board[pile[i-3]+1][i] == 2){
                                board[pile[i-3]][i] = 4;
                                board[pile[i-3]][i-1] = 4;
                                board[pile[i-3]-1][i-1] = 4;
                                board[pile[i-3]-1][i-2] = 4;
                                counter = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
```

```
                                if(counter == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]][i-1] = 9;
                                        board[pile[i-3]-1][i-1] = 9;
                                        board[pile[i-3]-1][i-2] = 9;
                                }
                        }
                        if( hyouka[2][2] == 1 && hyouka[1][2] == 1 && hyouka[1][3] == 1 &&
hyouka[0][3] == 1 && board[pile[i-3]+1][i] == 2 && board[pile[i-3]][i+1] == 2){
                                board[pile[i-3]][i] = 5;
                                board[pile[i-3]-1][i] = 5;
                                board[pile[i-3]-1][i+1] = 5;
                                board[pile[i-3]-2][i+1] = 5;
                                counter = 0;
                                counter2 = 0;
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]][m] == 2)counter++;
                                }
                                if(counter == 18){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-1][i+1] = 9;
                                        board[pile[i-3]-2][i+1] = 9;
                                }
                                for( int m = 3 ; m <= nw-3; m++){
                                        if(board[pile[i-3]-1][m] == 2)counter2++;
                                }
                                if(counter2 == 17){
                                        board[pile[i-3]][i] = 9;
                                        board[pile[i-3]-1][i] = 9;
                                        board[pile[i-3]-1][i+1] = 9;
                                        board[pile[i-3]-2][i+1] = 9;
                                }
                        }
                        break;
                }
                }
        }
        public int pile(int x){
                int i;
                yy = 0;
                out:{
                for(i = 0; i <= nh-3; i++){
                        if(board[i][x] == 2){
                                yy = i-1;
                                break out;
                        }
                }}
                if(yy == 0) yy = 30;
                return yy;
        }
}
```