学士論文

C 言語による BlokusDuo 対戦プログラムの作成

氏　　名　：　古川　晋也
学籍番号　：　2260100051-6
担当教員　：　山崎 勝弘 教授

提 出 日　：　2014 年 2 月 20 日

立命館大学　理工学部　電子情報デザイン学科

内容梗概

近年では、人工知能を用いた製品が数多く存在している。人工知能とは、人間の知能そのものを持つ機械と、人間が知能を使ってすることを機械にさせることの二つがあるが、製品の人工知能は後者にあたる。また、製品以外でも、将棋、チェス、オセロなど、様々な対戦競技の分野で人工知能が使われ、人とコンピュータプログラムの対戦が行われている。

本研究の目的は本研究室ではインパルス C では作られているが、C 言語ではまだ基礎までしか作られていない BlokusDuo の対戦プログラムの作成である。本研究ではそのプログラムにルールを付け加えることにより、人対人の対戦ができるようにした。

本論文では、BlokusDuo のルールについて説明した後、作成したルールのプログラムのアルゴリズムについて説明し、最後にそのプログラムのシミュレーションによる動作検証を行う。

目次

図目次

１．　はじめに

近年では、人工知能を用いた製品が数多く存在している。人工知能とは、人間の知能そのものを持つ機械と、人間が知能を使ってすることを機械にさせることの二つがあるが、製品の人工知能は後者にあたる。身の周りの人工知能の例をあげていくと、まず１つ目として、自動掃除ロボットなどがある。これはセンサーにより掃除していくのだが、壁端のゴミをかき出す、テーブルの足回りをくるくる回って掃除する、ゴミが多いところは念入りにするなどなどをセンサーにより見極めて最適な行動を、毎秒 60 回以上も選択するというものである。また iPhone に搭載されている Siri などもそうである。Siri とは、認識した言葉に反応して、ユーザーが求めるアプリケーションを起動したり、質問に回答したりすることができるものである。これは自然言語処理により様々な言葉を理解し、柔軟な検索技術で回答を検索・作成し、音声合成により回答するものである。

また、特に注目していただきたいのがゲームでの対戦プログラムである。最近では、将棋、チェス、オセロなどのいろいろな分野で人とコンピュータプログラムの対戦が行われている。その中の１つとして、プロ将棋とコンピュータ将棋プログラムの対戦が話題になってきている。コンピュータ将棋は 1970 年代から作られ始めたが、当時のプログラムの強さはとても弱く、アマチュア初段の人がハンデつきでも楽々勝てるほどであった。しかし、念を追うごとにプログラムの強さは上がって行き、2005 年の「激指」、2006 年の「Bonanza」の登場からプログラムの強さは劇的に変わった。ほかのオセロや囲碁、将棋などでもプロ対コンピュータは行われており、今ではコンピュータがプロに勝つ時代になってきている。全ての科目でコンピュータが人を超える時代はそう遠くはない。また、これらはネット上などでもその様子が動画として配信されていたりもする。

本研究室では、BlokusDuoの対戦プログラムがつくられている。BlokusDuoとは、2000年にフランスのSekkoia社より発売された４人用のボードゲームを小型化し、二人用にしたものである。二人は自分のパズルを置いていきながら、相手のパズルを置くのを邪魔しあい、パズルの置けたパズルの種類や数を競い合う。このゲームは初手の可能手が
400 手近くもあり，序中盤でも可能手が800 手を超えることが多いため，序中盤においては深く探索するのは難しいとされる。しかし、終盤に移行するにつれ可能手は減っていくため、いかに末端まで読み切るかという問題になってくるのである。

この BlokusDuo は東京農工大学や相磯秀夫杯 FPGA デザインコンテストなどでプログラム同士の大会なども行われいる。また、本研究室の先輩方は相磯秀夫杯 FPGA デザインコンテスト、BlokusDuo 競技会で 3 位入賞するなどの実績もある。

本研究では、この対戦ゲームを C 言語で記述し、人とコンピュータが対戦できるようにするのが目的である。本研究室では、インパルス C による記述の BlokusDuo 対戦プログラムはすでにあるが、C 言語で記述された対戦プログラムはまだ基礎の部分までしか作られていない。なので、まだ作られていない言語でも対戦プログラムを作ることにより、今よりも幅広い言語で作れるようにし、より汎用性を高める。また、作成したプログラムをシミ

ュレーションしてみることにより、正しく動作しているかを確認する。

本論文では、第２章で BlokusDuo のルールについて述べる。第３章では作成した対戦プログラムの機能やアルゴリズムについて述べる。第４章では、対戦プログラムの機能とアルゴリズムについて述べる。第５章では、実際にシミュレーションしてみたことについて述べる。

２．　BlokusDuo のルール

本章ではブロックの種類を紹介し、つぎにピースの初手位置を説明し、さらにピースの置きルールを説明し、最後に勝敗の判定について説明する。

2.1　ブロックの種類

まず２人のプレイヤーはそれぞれ橙色のピースと紫色のピースに分かれ、持ち駒として小正方形が１～５個繋がった形のピースを持つ。ピースの種類は 21 種類となっている。プレイヤーは各種のピースを１枚ずつ持っており、計 21 枚が持ち駒となる。ピースの種類は図１のとおりである。

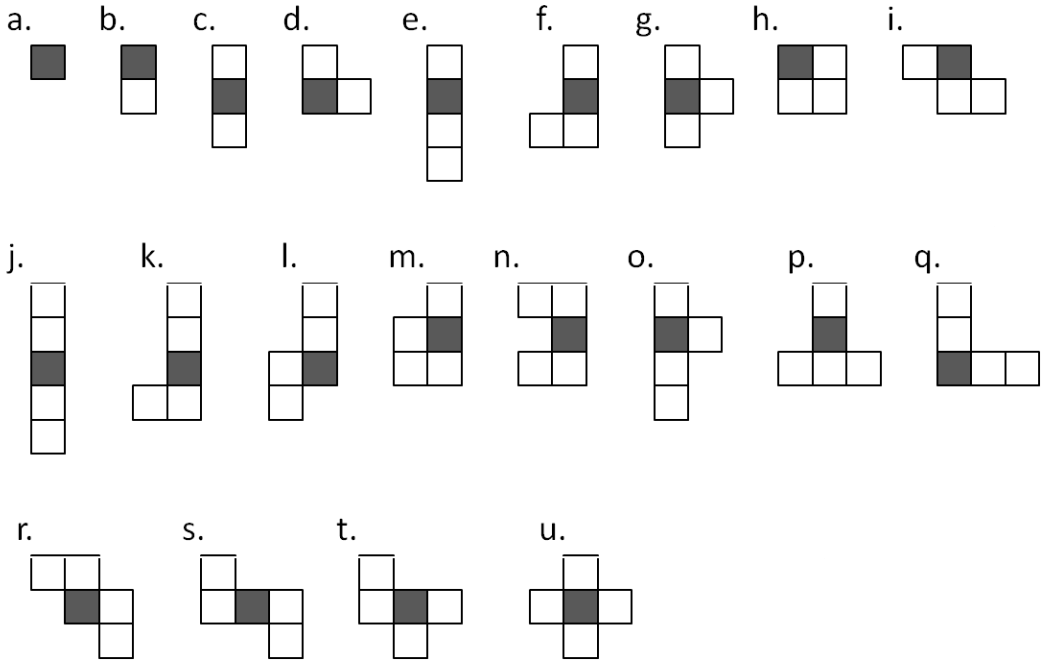

図１　ブロックの種類

2.2　ピースの初手位置

このゲームは14×14のマスに区切られた図２に示すボード盤にパズルを交互に置いていく。このとき、まずスタート地点として、最初におかなければならない場所が決められている。右への軸をｘ軸、下への軸をｙ軸とし、1～14 までマスに数字とローマ字を振り当てたとき、マスの場所を[ｘ、ｙ]と表記するとスタート地点は[5,5]と[a,a]の二箇所となる。その状態を図２に示す。図２の×印をつけた場所がスタート地点であり、各プレイヤーははじめにその初期位置に重なるようにピースをおかなければならない。

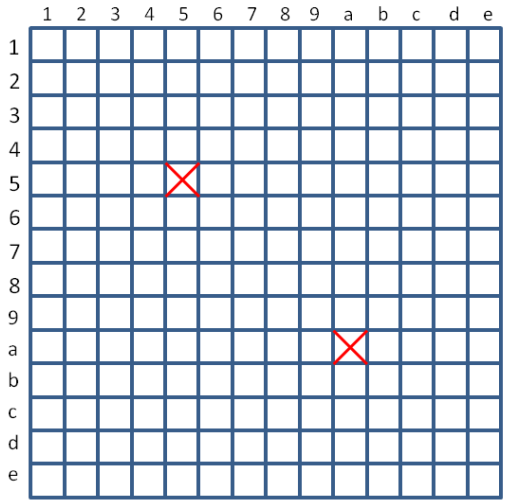

図２　ピースの初手位置

2.3　パズル配置の例

初期位置にお互いがピースを置くと、交互にピースを置いていく。この時、置いていくためのルールとして、BlokusDuo ならではのルールがいくつかある。

（１）　パズルとパズルが重なってはいけない。
（２）　自分のピース同士が頂点で接するように置かなければいけない。
（３）　自分のピース同士が辺で接してはいけない(他者のピースとは辺で接しても良い)。

この３点が挙げられる。これらをわかりやすくするために図３に良い例悪い例を示す。

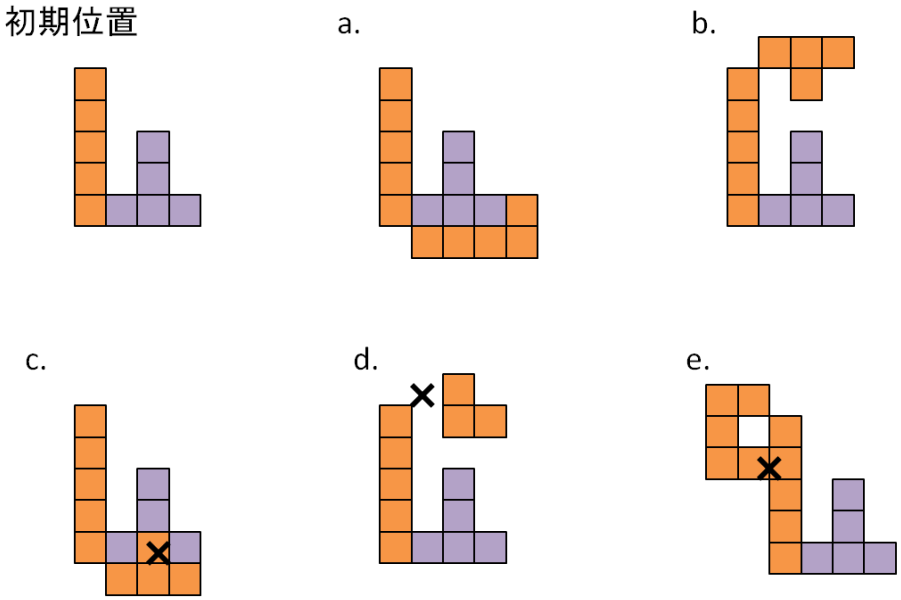

図３　パズル配置の良い例悪い例

この場合、ａとｂは全てのルールを見たしており良い例となる。ｃｄｅは悪い例であるが、ｃの場合はパズル同士が重なってしまっているため置くことが出来ない。ｄの場合は角と角が接しているところがないため置くことが出来ない。ｅの場合は、ハント辺が接してしまっているため置くことが出来ない。という風になる。

2.4　勝敗の決め方

交互にピースを置いていき、どちらも置けなくなるまでゲームを行い、両方とも置けなくなったらゲーム終了。この時点で残っているピースのコマの合計マス数の少ない人の勝ちとなる。採点の例として、図４に残りの駒を示す。

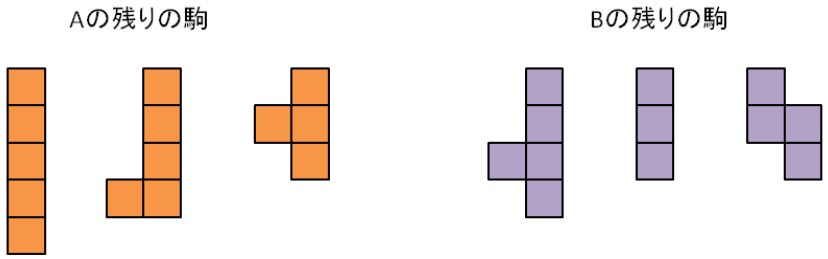

図４　双方の残りの駒

Ａと Ｂがこのように持ち駒が残った場合、Ａのコマの合計数は、５マス×２と４マス×１なので、１４となる。また、Ｂのコマの合計数は、５マス×１と ４マス×１と３マス×１で、１２マスとなる。そのため１４対１２となり、マスの合計数の少ない Ｂ の勝ちとなる。

３．　ルール作成のアルゴリズム

ここでは、ピース同士が重ならない、自分のピース同士が頂点で接するようにおく、自分のピース同士が辺で接さないようにするなどのルールがどのようなアルゴリズムでできているかについて述べる。

3.1　判定するためのマス目の作成

まず、自分の置いたピースによって置く場所が制限されるルールは、上記で示したもの全てにあたる。しかし、相手の置いたピースによって制限されるのはピース同士が重ならないという一点のみである。その為、同じ盤面でも自分の置ける場所と相手の置ける場所は異なってくる。なので、その場所が置くことができるかどうかを判定するためのマス目を２つ作り、自分側がおけるか判定するためのマス目を判定 A、相手側がおけるかを判定するためのマス目を判定 B としておく。このプログラムを図 5 に示す。

```
int xbox[14][14];
int ybox[14][14];

for(i=0;i<14;i++){
        for(j=0;j<14;j++){
                xbox[i][j]=0;
        }
}
for(i=0;i<14;i++){
        for(j=0;j<14;j++){
                ybox[i][j]=0;
        }
}
```

図 5　判定マス目の作成

これにより、１４×１４のマスが２つ作られ、その中のマス全てに０が入る。この状態を図 6 に示す。また、マス目の中身は初期は全て０が入っているものとするが、ここでは見やすくするために０は省略しておく。

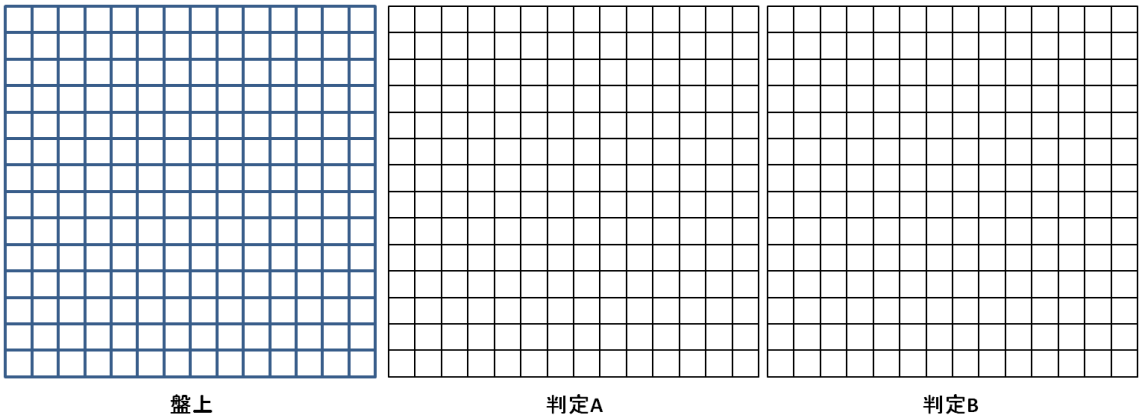

図 6　判定のマス目

3.2 置いてあるピースの判定

まずこちら側がピースを置いた場合、相手側の判定のマス目には盤上にこちらがわが置いたピースと同じ場所に 1 の判定を入れる。そして、自分の判定側にはピースと同じ場所と、辺が接している部分に 1 の判定を入れる。次に、置いたピースの角と接している場所に 2 を入れる。相手側がピースを置いた場合も同じようにする。この状態を図 7 に示す。また、ピースを置いた場所に入れる 1 の判定の色を黒色、ピースの辺に接している部分に入れる 1 の判定の色を緑色、角に接している部分に入れる 2 の判定の色を赤色としておく。

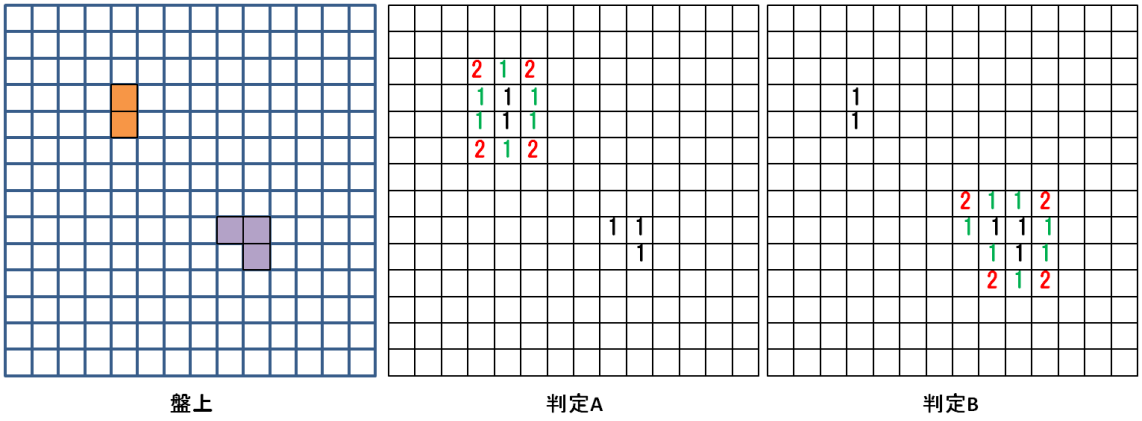

図 7　ピースの判定

3.3　ピースを置くときの判定

次に、2 つ目以降のピースを置く条件について説明する。ピースを置くときまず初めに、自分の判定のマス目のピースを置く場所に判定 1 がないかを確認する。もし 1 つでも 1 があれば、相手側か自分側のすでに置いてあるピースに重なっているもしくは自分の置いて

あるピースと今からおくピースの辺同士が接してしまっているということになる。よって、置く場所の全てが０か２であれば条件クリアとなる。

そして次に置いたピースの角になる部分に１つでも２が含まれているかを確認する。これはBlokusのルールである角と角が接しているように置くという条件を満たすためのものであり、3.2 でピースの角と接している部分に２の判定の判定を入れて置いたため実現することができる。もしピースの角の部分に１つでも２が含まれていれば条件クリアとなり、置くことができる。例として図 8 にピースの角の部分がどこであるかをいくつか示す。また、図 7 の状態から A が置いてあるピースの右下に５マスのピースを置く場合を図 9 に示す。

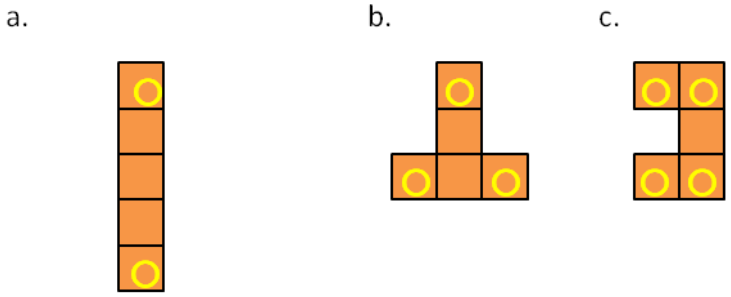

図 8　角の判定

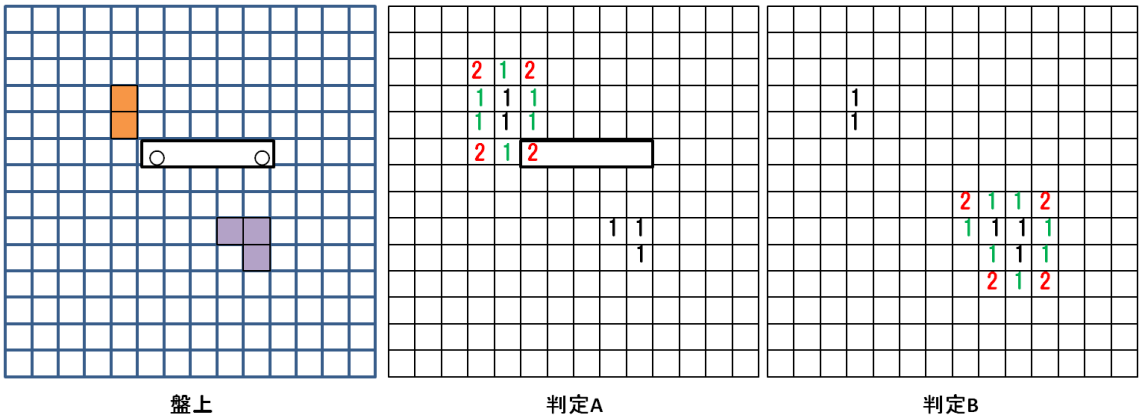

図 9　ピースの判定確認

図 9 の盤上のようにピースを置きたいとき、判定 A を見てみるとピースのを置く部分の判定には０と２しか入っていない。そして図 8 の a を参考に、ピースの角である一番左端の部分に２の判定が入っているため条件が全てクリアできているので、ここに置くことができる。

そしてそこにピースを置くとき、前述で説明したように判定に１と２を入れていくのだ

が、１を入れる場合はその場所に０と２のどちらがあったとしても問題なく入れていく。しかし２を入れていく場合、その場所の判定が０ならば２を入れるのだが、もし１が入っていた場合はそのまま１の状態にしておく。この状態を図 10 に示す。

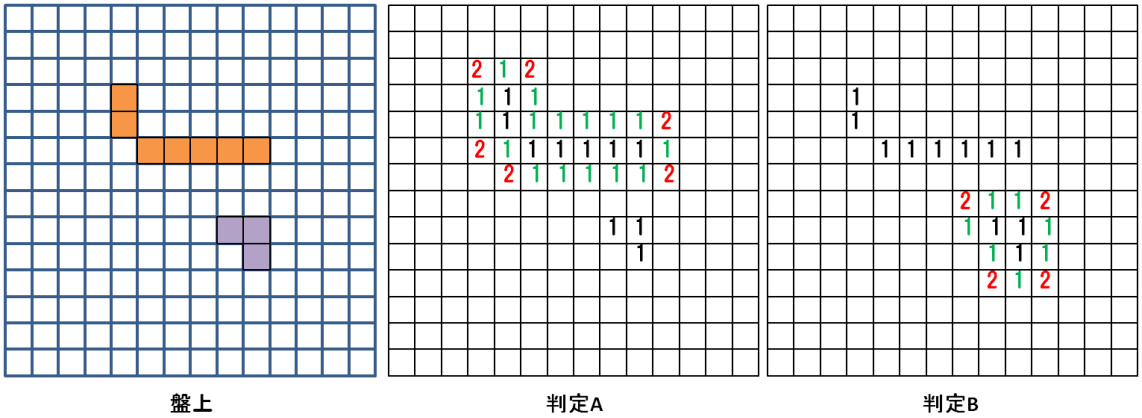

図 10　ピースと判定の入力

そして相手のターンになると、今度は判定Ｂの方で先ほど言ったことと同じ手順で条件をクリアしているかを確認し、もしクリアしていれば、判定Ａにはパズルの位置に１を、判定Ｂにはパズルの位置と辺に１を、角には２を入れていく。これを繰り返し、最終的に余ったパズルの種類と数を数え、勝ち負けを決める。

４．　対戦プログラムの検討

4.1　対戦プログラムの機能

ここではこの BlokusDuo の対戦プログラムの機能について説明する。現在、C プログラムの言語で書いたプログラムでできるのは人対人の対戦のみであるが、人対コンピューター、コンピューター対コンピューターについては今後どのような機能を付ける予定であるため、これについて説明する。

（1）　初期設定

最初に入力によって何対何の対戦かを決める。１を入力すれば人対人となり、２を入力すれば人対コンピュータの対戦となるようにする。コンピュータとコンピュータの場合は２つのプログラムに、人対コンピュータの２と入力する。なぜそのようにするかはあとに説明する。そして、２を入力した場合は人側が先手か、コンピュータ側が先手かを入力する。人側が先手の場合は１を入力し、コンピュータ側が先手の場合は２を選択する。この順序を図 11 に示す。

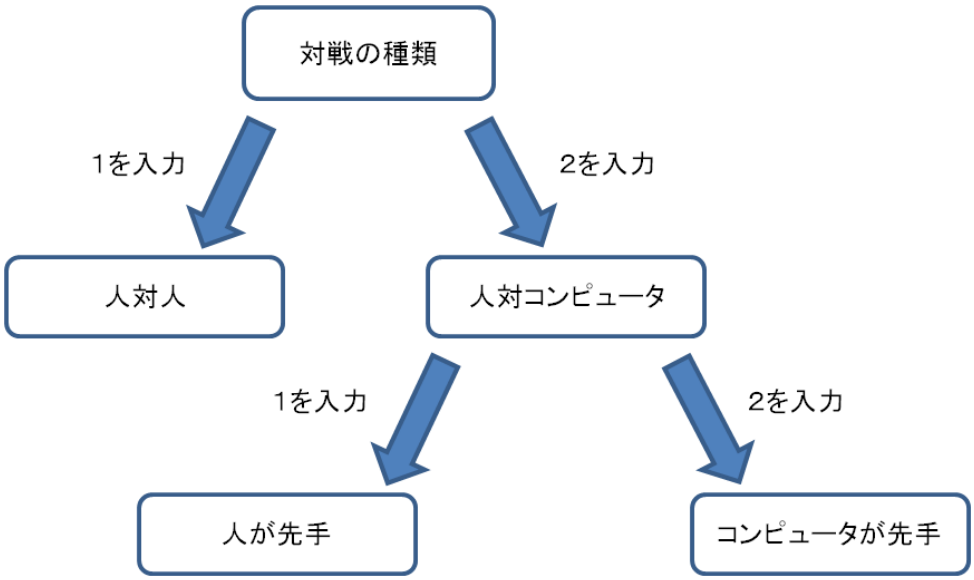

図 11　対戦形式の手順

（2）　盤上への打ち込みコード

そして次に順番に交互にピースを打ち込んでいくのだが、打ち込む方法として英数字４文字を入力する。この 4 文字のうち、1 文字目と 2 文字目はピースを置く座標を表し、3 文字目はピースの種類を表す。そして 4 文字目はピースの向きを表す。

(A)　座標

まず 1,2 文字目の座標を決める部分であるが、ピースには全ての種類に回転させる時の目

安になる中心部分がある。これは図1に示されているそれぞれのピース灰色の部分である。1文字目と2文字目の座標は、この灰色の部分がどの位置にあるかを示している。図2の通り、盤面には1～eの番号が振り分けられており、1文字目にはx座標、2文字目にはy座標のを指定する。例えば、1文字目と2文字目にそれぞれ「8,a」と打ち込めば、左から8番目上から10番目の座標となる。

(B)　種類

次に3文字目のピースの種類であるが、これは図1に振り分けられているとおりである。選びたいピースをa～uの中から選ぶ。

(C)　向き

そして最後に4文字目のピースの向きは0～7の数字のいずれかを入力する。この時、0を入力すればピースは図1の形の通りの向きとなる。そして1を入力すれば0のy軸反転の形となる。2は0ピースの形を時計回りに90°回転させ、3は2のピースをx軸反転させたものとなっていく。そして4以降も同じ順序で反転、回転されていく。例として、図1のlのピースが数字によってどのように向きを変えるかを図12に示す。

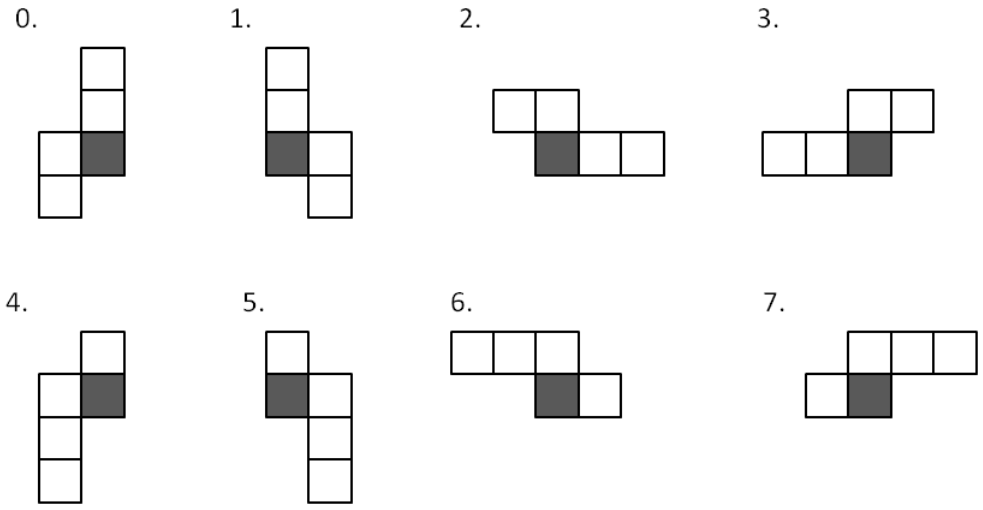

図12　ピースの方向

また、ピースの置ける場所がなくパスをするときは、4文字の入力の部分に「0000」と入力する。そして、もうピースが置き終わり、ジャッジするときには「xxxx」と入力する。ジャッジの部分では、盤面にある自分のピースの和人相手のピースの数を数えていき、盤面に置けたピースのマスが多い方が勝ちと画面に出てくる。

また、入力した際に置くことができない場合であれば、「すでにそのパズルは使われている」「その場所はピースを置けない」などのエラーメッセージが出るようになっている。

4.2　対戦プログラムのアルゴリズム

ここでは、対戦プログラムがどのようなアルゴリズムで処理されていっているかを各々の対戦内容ごとに説明していく。対戦プレイヤーが何と何であるかによって、対戦進行の処理のされ方は少しずつ変わってくる。

（1）　人対人の対戦

人対人の対戦の場合、先手か後手かを決めるのはプログラム上での入力ではなく、話し合いによってである。そして先手側がパズルの種類、向き、座標を打ち込む。そしてその入力が、BlokusDuo のルールにちゃんと沿っているかどうかを判別し、条件を満たしていなければ再びパズルの入力に戻る。条件を満たしていた場合、パズルはそこに置かれ、次の後手側のターンとなる。そして相手も同じようにパズルの入力をし、条件を満たしていれば先手側ターンとなり繰り返していく。この順序を図 13 に示す。

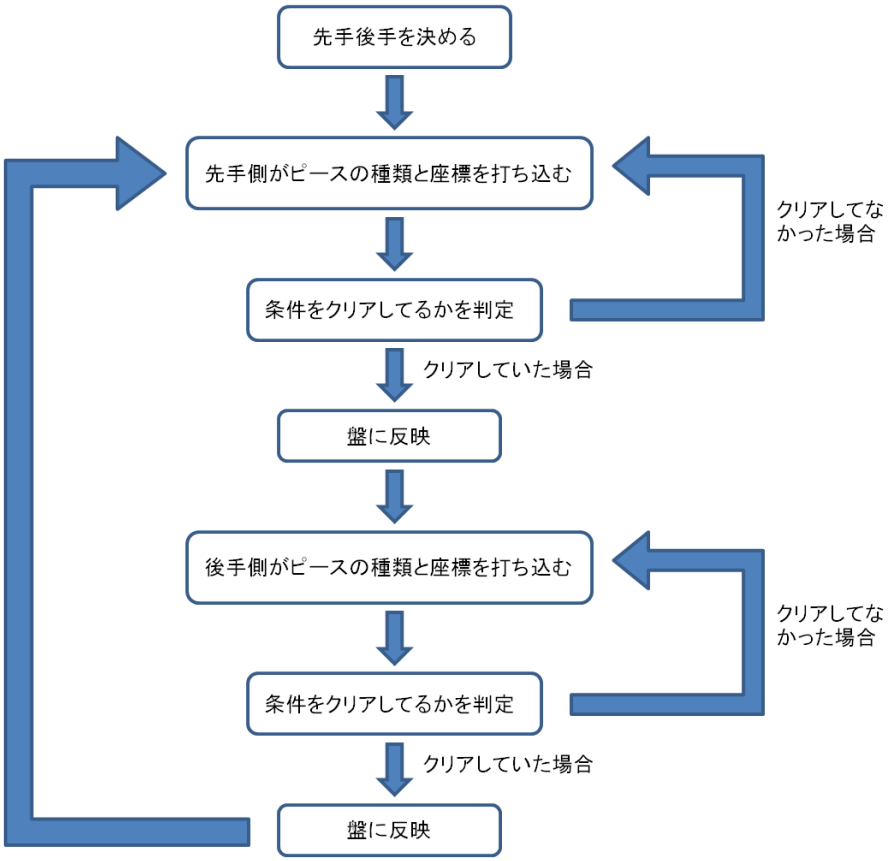

図 13　人対人の対戦アルゴリズム

（2）　人対コンピュータの対戦

人対コンピュータの場合、人がまず先手か後手かを入力によって決める。そして先手と入力した場合、まず人側がパズルの種類、向き、座標を打ち込む。そしてその入力が、BlokusDuo のルールにちゃんと沿っているかどうかを判別し、条件を満たしていなければ再びパズルの入力に戻る。条件を満たしていた場合、パズルはそこに置かれ、次のコンピュータ側のターンとなる。そして、コンピュータがどの場所に何を置けばいいかを探索し、結果が出ればパズルを盤上に打ち込む。このとき、コンピュータのターンでは元から置けるところにパズルを置くのが前提に、そのプログラムが最もよいと思う場所にピースを置くため条件をクリアしているかどうかの判定は行われない。この流れを図 14 に示す。

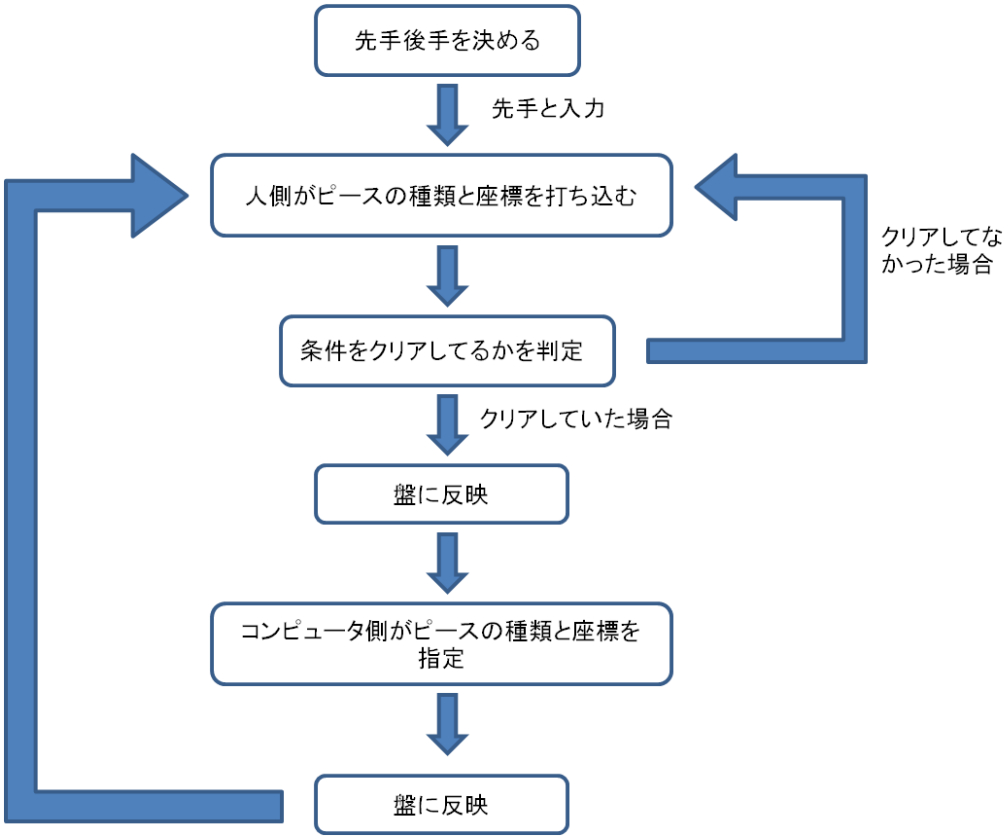

図 14　人対コンピュータの対戦アルゴリズム

（3）　コンピュータ対コンピュータの対戦

コンピュータ対コンピュータの場合、まず先手か後手かを決める。両方人対コンピュータの設定にし、片方には先手、もう片方には後手と入力する。先手側がどの場所に何を置けばいいかを探索し、結果が出ればパズルを盤上に打ち込む。そしてその打ち込んだ手を、

後手側は人が打ったものとして入力する。それによって出た後手側の手を、今度は先手側に人が打った手として入力する。それを繰り返していくことにより、ゲームを進行させていく。この流れを図 15 に示す。

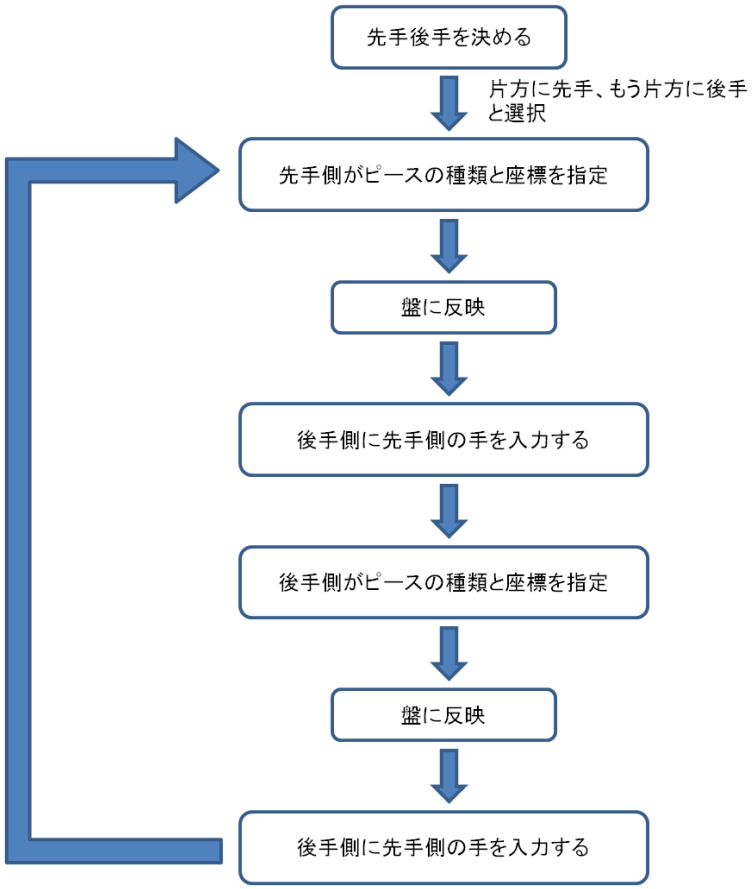

図 15　コンピュータ対コンピュータの対戦アルゴリズム

（4）　ゲームの終盤のアルゴリズム

ゲームが終盤になっていくにつれ、片方のピースが置けなくなってくる。そうなると置けなくなった方はパスをし、もう片方は自分が置けなくなるまでパズルをおいていき、最終的にどちらも置けなくなると勝敗の判定に入る。後手側が先に置けなくなった場合のこの時の状態を図 16 に示す。

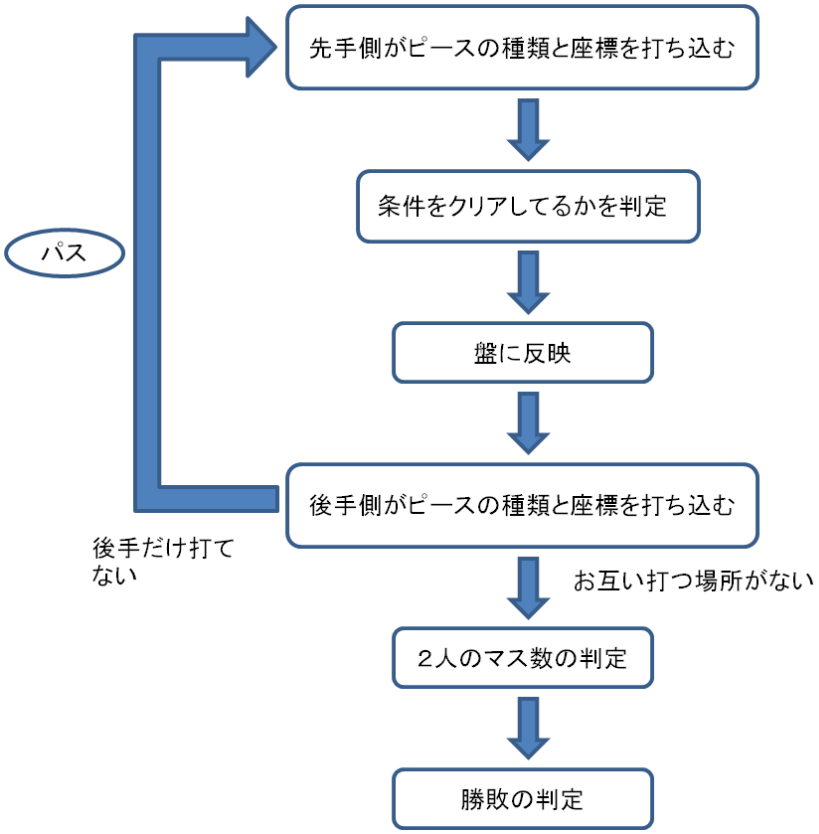

図 16　ゲーム終盤のアルゴリズム

５．　シミュレーションによる動作の検証

5.1　シミュレーションの動作の条件

ここでは、シミュレーションをするにあたりどのような条件で実行したかを次のとおりに示す。このプログラムは C 言語により作られた。また、Cygwin のによりプログラムを起動した。対戦相手は人対人で行うまた、先手後手はコンピュータとの対戦でないため考えないものとする。ピースがまだ置ける場所があってもパスをすることはできるものとする。

5.2　シミュレーション結果

（1）　ゲームの序盤

プログラムを起動したときの初期状態を図 17 に示す。

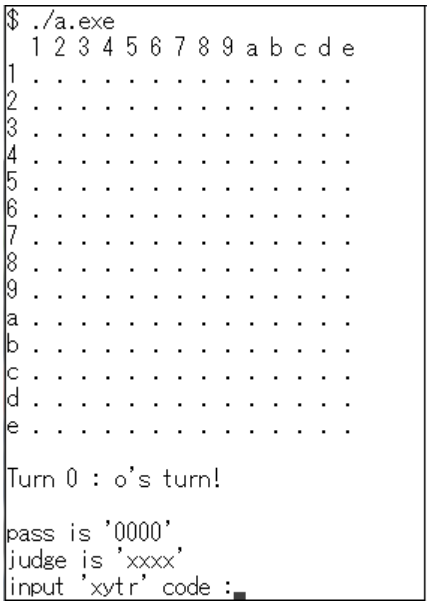

図 17　シミュレーションの初期画面

まず、横に並んでい 1 ~e までの数字が盤上の x 軸を示し、縦に並んでいる 1 ~e までの数字が盤上の y 軸を示す。盤上の下に表示されている Turn 0：○'s turn!という表示により、今のターンと先手と後手どちらが次に打つ番加賀わかる。また、先手が○、後手が×となっている。そして「input 'xytr' code」というところに 4 文字の入力をすることにより、盤上にピースを置くことができる。この xytr とは、xy が x 軸 y 軸をを示し、t がピースの種類を示し、r がピースの向きを示している。「pass is '0000'」と表示されているのは、code に 0000 と入力されれば入力した側がパスになるということである。「judge is 'xxxx'」表示されているのは、code に xxxx と入力されればゲームが終了し、勝敗の判定にうつるということである。

また、1 ターン目 2 ターン目の流れを図 18、図 19 に示す。

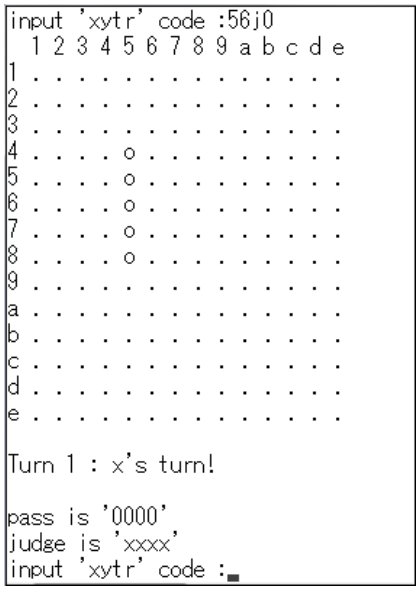

図 18　1 ターン目

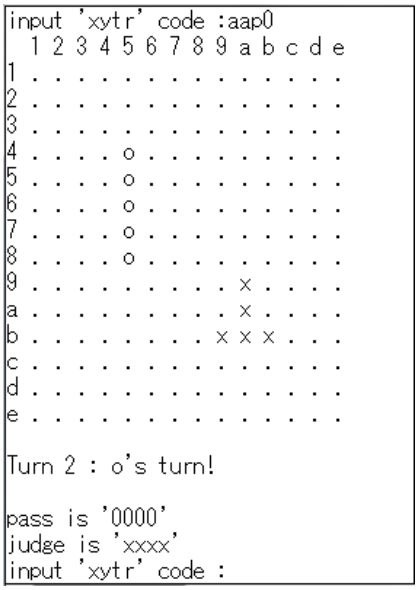

図 19　2 ターン目

まず 1 ターン目に先手が code に「56j0」と入力したことにより、図 18 の盤上に○で表示された部分にピースが置かれたということになる。また、2 ターン目に後手が「aap0」と入力したことにより、図 19 の盤上で×で表示された部分にピースが置かれたということになる。

（2）　ミス配置時のエラーメッセージ

また、すでに使っているパズルを使ってしまった場合のエラーメッセージと、ピースが被った場合のエラーメッセージを図 20、図 21 に示す。

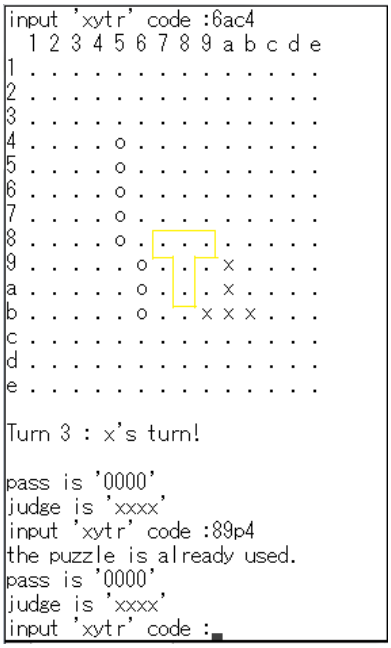

図 20　同じパズルを置いた場合

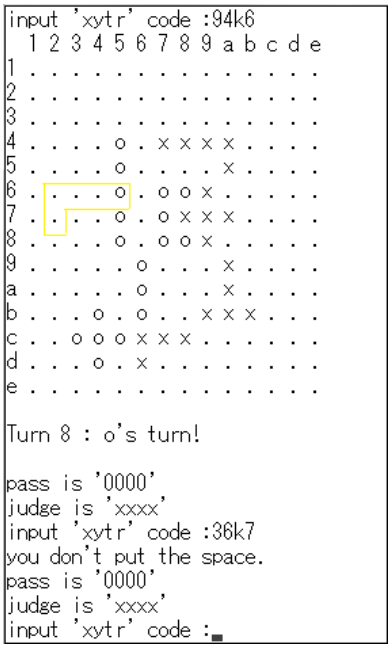

図 21　パズルが重なった場合

図 20 では後手の×側が code に「89p4」と入力したことにより、画面の黄色で囲われているところにパズルがおかれようとした。しかしこの形のパズルはすでに置かれているため、入力の下に「the puzzle is already used.」と表示され、再度 code を入力する場面に戻った。また、図 21 では先手側の○code に「36k7」と入力したことにより、黄色で囲まれている部分にパズルが置かれようとした。しかしこの部分はパズルが他のパズルに重なってしまっているため、「you don't put the space.」と表示され再度 code を入力する画面に戻った。。

（3）　デバッグの最中

また、ゲームを進行していると、本来置けるはずの場所に置くことができなかった。その様子を図 22 に示す。

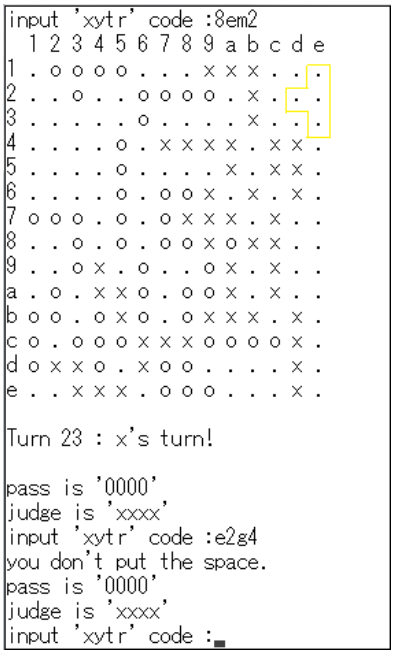

図 22　プログラムミス

本来 code に「e2g4」と入力したことにより、図 22 のように黄色で囲われた部分に×が入るはずであった。しかし何かの問題により、本来はパズルが重なったりしたときに出るはずのエラーメッセージである「you don't put the space.」という表示が出てしまい、置くことが出来なかった。しかし別の形のパズルで試してみると、置くことができた。

（4）　ゲームの終盤と勝敗の判定

また、このままゲームを進行していき、どちらもパズルを置けなくなった状態を図 23 に示す。そして、その後の勝敗の判定を図 24 に示す。

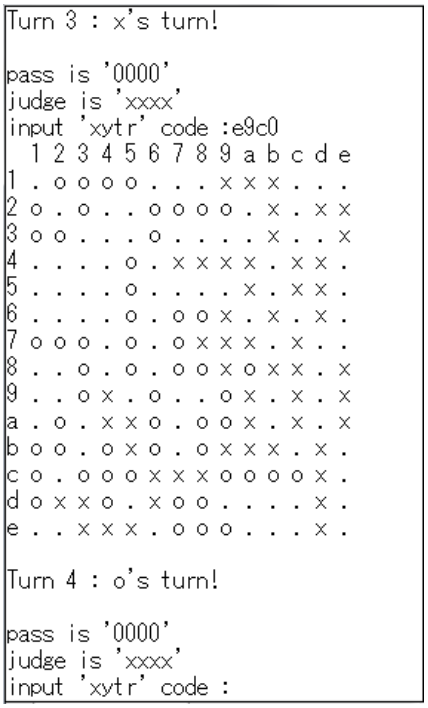

図 23 最後の状態

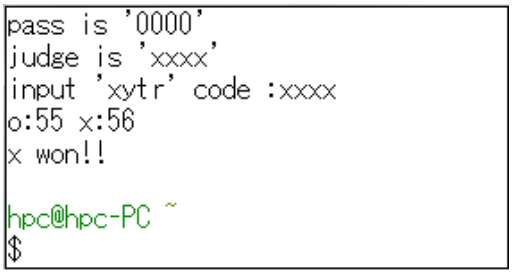

図 24　勝敗判定

この図 23 を見るように、code に xxxx と入力することにより「○：59　×：56」と表示された。これは図 22 で表示されている盤上に○が 59 個、×が 56 個あるということになる。つまり置けたマスが多いということは、余っているピースのマスの数が少ないということなので、「○ won!!」と書かれているように、○の勝ちという流れになる。

（5）　パスをした場合

また、パスを行った場合どのような処理になるかを図 25 と図 26 に示す。

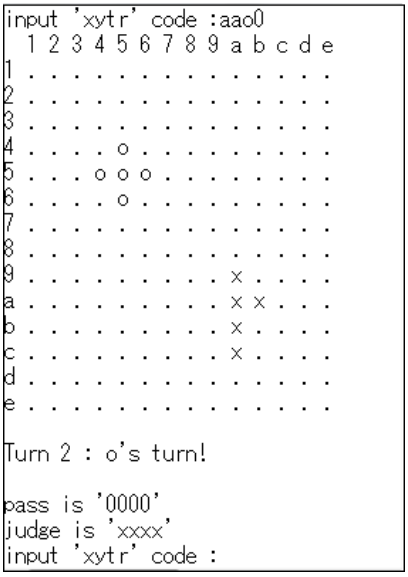

図 25　パスする前

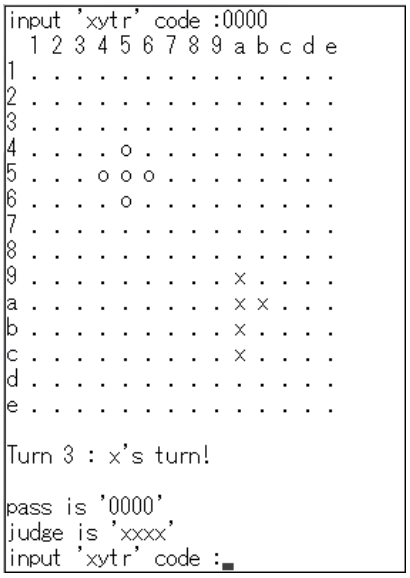

図 26　パスした後

このように、図 25 の場面で code に「0000」と入力することにより、図 26 の状態になる。図 25 では Turn2 で先手側の○のターンであったが、図 26 では Turn3 で後手側の×のターンとなっている。これにより、ターンが 1 進み、先手後手も入れ替わったことによりパスがちゃんとされていることがわかる。

5.3　考察

今回、基礎がかかれていたが、まだパズル同士が重ねて置けてしまったりする既存のプログラムに同じピースを 2 回使えなくする、パズル同士が重ならないように置く、自分のピースの辺と辺が合わさらず、角と角が合わさるように置く、などのルールを付け加えた。その結果図 21 のようにおけるはずの場所に置けないというようなことが起きてしまった。これはおそらくピースを上端、もしくは下端においてしまうことにより、反対側の部分のマス目に判定の 1 などを入れてしまうからだと思われる。この問題を解消するためにはマス目を 14×14 ではなく、15×15 などに増やしてやれば良いと思われる。また、本研究室ではインパルス C ですでに BlokusDuo の対戦プログラムが書かれていたが、同じものを作るにしても C 言語で書くとインパルス C で書くよりもかなり多く書かなければならなかった。これは、もともとインパルス C が対戦プログラムなどを作るのに向いている言語であるからと思われる。

６．　終わりに

本研究では、C 言語による BlokusDuo の人対人の対戦プログラムを作った。人対人でのシミュレーションと、動作検証を行った。それにより、人対人でのプログラムが一部を除き正常に動作することが確認できた。また、インパルス C の方が BlokusDuo の対戦プログラムに向いていることがわかった。

今後の課題として、今回の研究では人対人までしか作れなかったプログラムを、人対コンピュータの対戦までできるようにプログラムを完成させることがあげられる。

謝辞

本研究の機会を与えてくださり、ご指導をいただきました山崎勝弘教授に深く感謝いたします。また、本研究に関して貴重な助言、ご意見またきっかけをいただきました孟林助教授、杵川大智様、そのほか高性能計算研究室の皆様に心より感謝いたします。

参考文献

［1］柴田望洋：新版 明解 C 言語 入門,SoftBank Creative,2004

［2］コンピュータ将棋協会：人間に勝つコンピュータ将棋のつくり方,技術評論社,2012

［3］杵川大智・石川陽章・孟林・山崎勝弘：FPGA ボードを用いた Blokus Duo 対戦アルゴリズムの設計と実装　電子情報通信学会 2013

［4］築地毅・大崎泰寛・酒井香代子・藤波順久・小谷善行：**コンピュータブロックスデュオ大会報告**：
http://www.tuat.ac.jp/~kotani/docs/GI22-4.pdf