2001 年度 卒業研究

オンラインゲームについて

岡山理科大学
総合情報学部
数理情報学科

－澤見研究室－

I98N001　赤池吉雄
I98N059　佐藤泰弘

目次

1. はじめに

オンラインゲーム作成を考える時,ゲームのアイデアや画像データなどのデザインは別にして,初めてのプログラム作成で最も苦労するのはデータ通信をどのように実現するかということであろう.そしてネットワークを介してデータやメッセージをやり取りする場合に考えなければならないのは通信プロトコルについてになる.通信プロトコルには各種のものがあるが,インターネット接続されたコンピュータ間でデータ通信を行う場合,ローカルまたはリモートに限らず,最も良く使用されている通信プロトコルはTCP/IPだと考えられる.そして,このプロトコル上でゲーム用データをやり取りする場合,これに適したプロトコルの細部をどうするかについて考える必要がある.そしてこの場合,各種ネットワーク環境で互換性のある動作を可能にしようとすると,固有のポート番号を割り当てるなどして,ゲームのデータを埋め込むのに適したプロトコルをさらに埋め込むなど多くの複雑な処理を必要とすることになる.そこで私たちはプロトコルに大きく依存せずオンラインゲーム作成が行えると考えられたDirectX を使用し,簡単なオンラインゲームを作成することにより,データ通信を行う上での要点を調べ,これを紹介しようと考えた.そしてこれを適用して多くの人がデータ通信を身近に感じ,オンラインゲーム作成に興味を示してもらえればと思う.

2. DirectX について

2.1 DirectX とは

DirectX とは Microsoft Windows オペレーティングシステムに組み込まれている複数の API(Application Programming Interface)のことである. 豊富な機能を有する DirectX を用いることにより,開発者は直接コードを書くことなくハードウェアにアクセスすることができるので,システムの提供するリソースを自由に使用することができる.このようにして DirectX は Windows ベースの PC に対する標準開発プラットフォームを提供している.DirectX は 1995 年に初めてリリースされ,Windows プラットフォームにおけるマルチメディア・アプリケーションの標準開発環境となっている. 現在 DirectX はバージョン 8.1 まで登場している.バージョンによる違いはプログラミングの細部に関してまで確認はしていないが,API に付けられた名前が少し違うということである.このことより DirectX は常に新しいバージョンを使用した方が好ましいと考える.今回の研究では DirectX8.1 でも互換性の問題は少ないと考え動作は確認できたため, DirectX7.0 を使用した.

2.2 DirectX 7.0 の API

DirectX7.0 で使用することのできる API を表 2.1 に示す.本研究ではオンラインゲームを作成するため,API の一つである DirectPlay を使用した.

API	説明
DirectDraw	2D 画像を作成する際に使用する
Direct3D Immdiate Mode	3D レンダリング機能のインターフェイス
Direct3D Retained Mode	リアルタイム 3D 画像をサポートしている
DirectSound	音を再生する機能をサポートしている
DirectMusic	MIDI 形式の音楽データを処理する機能
DirectPlay	ネットワーク対戦ゲームの開発をサポートする
DirectInput	ゲームコントローラなどの入力機器の処理を行う
DirectShow	画像や音声をストリーミング再生することができる
DirectTransform	アプリケーションを開発する為のツールである
DirectAnimation	色々なメディアを一つにし,アニメーションで表示する機能である

表 2.1　DirectPlay の API

2.3 DirectPlay とは

DirectPlay は各種データ通信のためのプロトコルを包含しており,モデム接続,TCP/IP 接続,IPX 接続,シリアル接続に対応している.さらにプログラマの選んだ通信方法を追加することもできる.この DirectPlay を用いることによって,通信サービスへのアクセス方式が容易になり,下位プロトコルやオンラインサービスに依存しない方式のオンラインゲームを作成することが可能となる. その結果どのようなネットワークを使ってマシン同士を接続するか,その際のプロトコルに何を選ぶか,ということをプログラマがあまり意識しないで済むようにすることができる.

DirectPlay は基本モデルとして,ピア・ツー・ピアおよびクライアント/サーバのどちらのデータ通信モデルにも対応している. また,ロビー通信モデルと呼ばれているモデルにも対応することができる.そして,どのモデルを用いるかをプログラマは意識する必要がある.そこで,各モデルの概略としてピア・ツー・ピア通信モデルを用いたプログラム例について述べることにする.他のモデルを用いた場合のプログラムについては,モデル間の差を意識することにより,同様にして作成することができるものと考える.

3. DirectPlay 基本モデル

DirectPlay を用いてオンラインゲームを作成する際に利用することができる通信モデルは,先に述べたようにピア・ツー・ピア通信モデル,クライアント/サーバ通信モデル,ロビー通信モデルの三つである.以下にこれらの通信モデルについて説明する.

3.1 ピア・ツー・ピア通信モデル

複数のマシンで同じプログラムが動き,データもそれぞれが同じものを有するようなモデルである.全てのデータを特定のユーザ以外の全ての参加者のマシンに対して直接送ることにより,データを全てのマシンで同一にするモデルである.プログラムは比較的簡単だが,参加者が増えるに従ってネットワーク上に流れるデータ量が急増する為,どちらかというと小人数向きのモデルといえる.

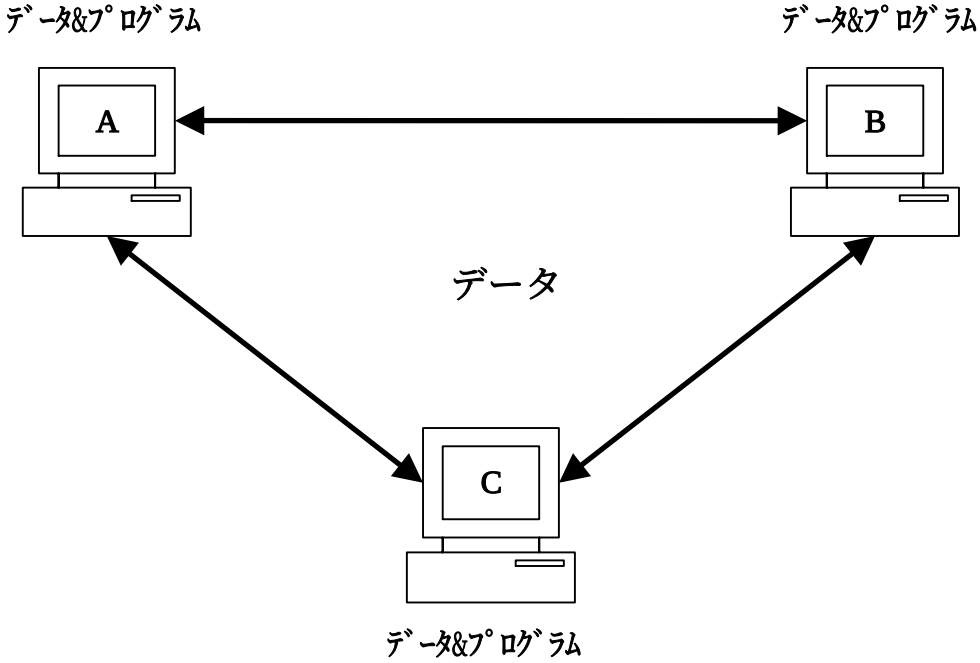

図 3.1 ピア・ツー・ピア通信モデル

上図に示した,ピア・ツー・ピア通信モデルの例では,クライアントＡでデータの変更が行われた場合,変更されたデータがクライアントＢとクライアントＣに送信されこれによりそれぞれのデータを更新する.クライアントＢでデータの変更が行われた場合,変更されたデータがクライアントＡとクライアントＣに送信される.クライアントＣでデータの変更が行われた場合,クライアントＡとクライアントＢに変更されたデータが送信される.どの場合においても.各クライアントは受け取ったデータを元にして更新を行うので,どのクライアントのデータも同一にすることができる.

3.2 クライアント/サーバ通信モデル

クライアント・プログラムがサーバ・プログラムと対話形式でデータをやり取りしてデータの整合性を保つ為の処理を進める通信モデルである.サーバ側のデータを共有しクライアント側のデータと組み合わせることで各クライアントのデータを同一のものにする.その結果,ネットワークに流れるデータ量が少くなる為,多人数にも対応しやすいが,プログラムは複雑になる傾向にある.

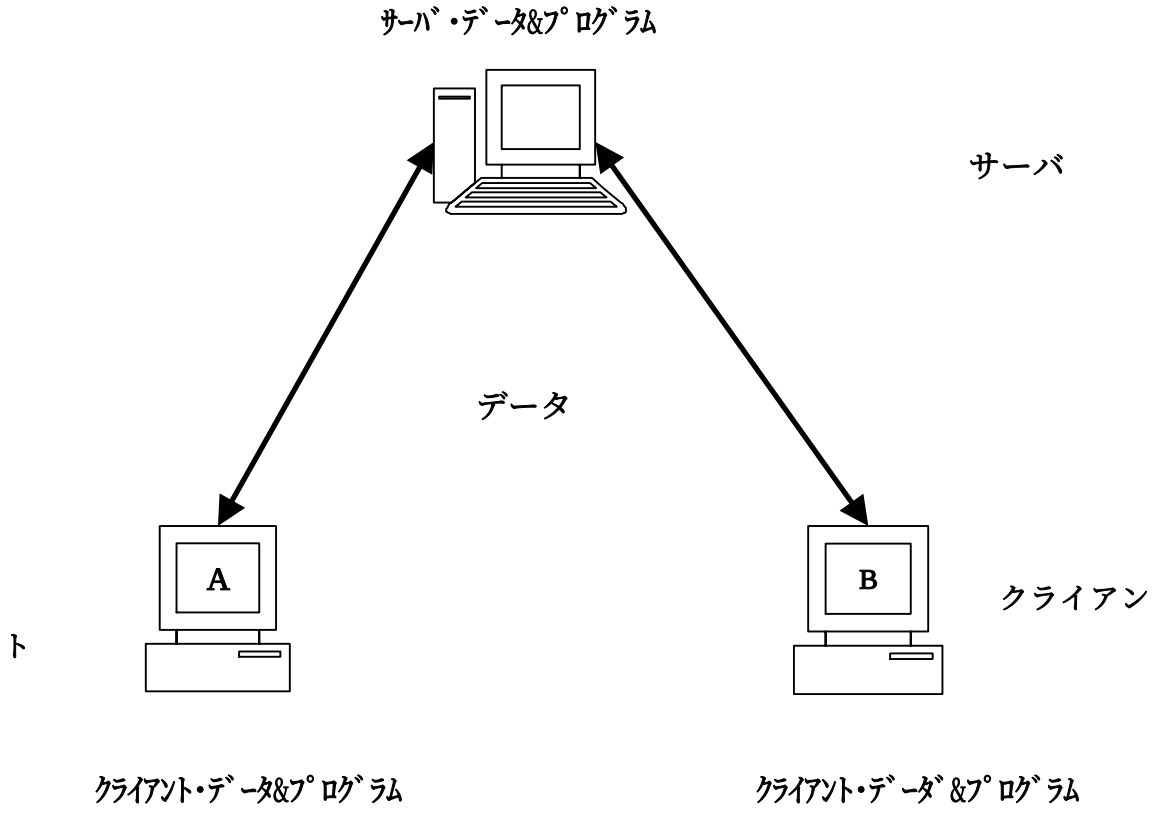

図 3.2 クライアント/サーバ通信モデル

上図に示したクライアント/サーバ通信モデルの例ではクライアントＡとクライアントＢでデータを変更した場合その内容をサーバに送信し,サーバはこの送信された情報を基に処理を行い,変更に必要するデータをクライアントＡとクライアントＢのそれぞれに送信する.これによりクライアントＡとクライアントＢのデータは更新され同一のものになる.

3.3 ロビー通信モデル

ロビー通信モデルとは,ロビー・サーバとロビー・クライアントによって成り立つ通信モデルである．ロビー・サーバとは,ユーザが集うネットワークの共通の場所でもある．ロビー・サーバが様々なプレーヤおよびセッションのデータ及びネットワークアドレスを管理しているので,ユーザはロビー・サーバを経由して興味のあるセッションを見つけたり,他のユーザとチャットしたり,セッションに参加したりすることができる．各クライアントに流れるデータ量はクライアント/サーバ通信モデルと差はないが,クライアントはロビー・サーバより多くの情報を得ることができる.その結果他のモデルに比べ,多くのサービスを実現することができる.例えば,トーナメントや個別のスコアを表示したり,掲示板を実装したり,会員登録制にすることもできる.

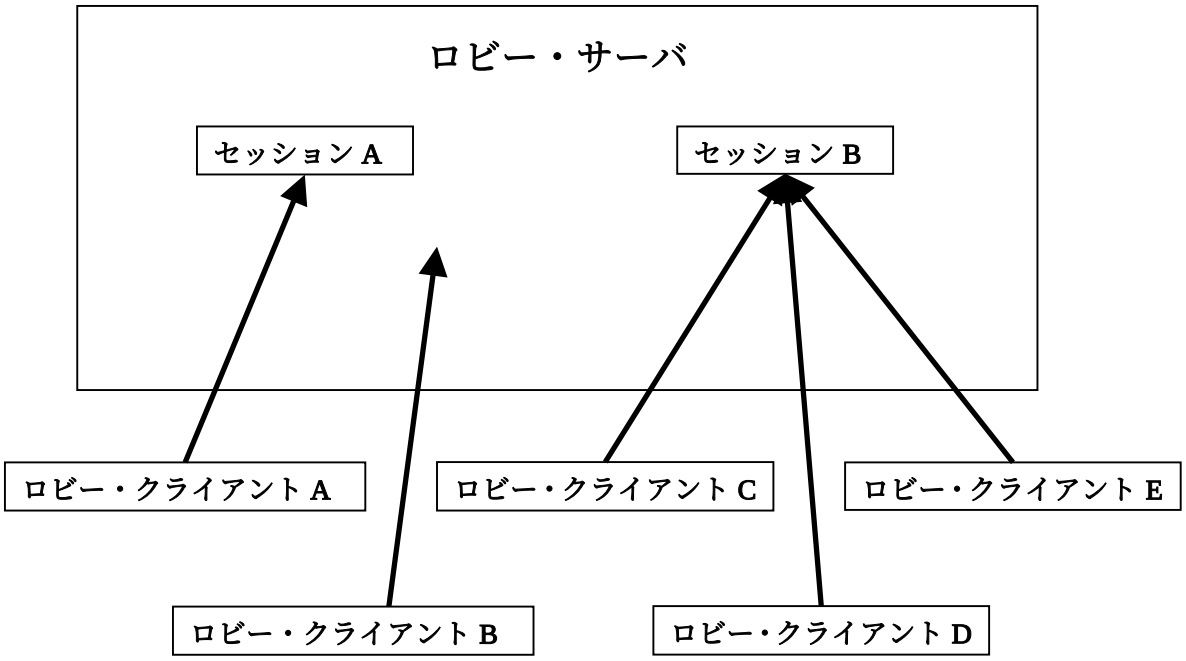

図3.3　ロビー通信モデル

上図に示したロビー通信モデルの例では,全てのロビー・クライアントがロビー・サーバと接続している状態を示している.ロビー・クライアントBはロビー・サーバと接続してセッションを探している状態である.ロビー・クライアントAは新規に作成したセッションAに参加している状態である.ロビー・クライアン

トＣとロビー･クライアントＤとロビー・クライアントＥはセッションＢに参加している状態である.

4 ピア・ツー・ピア通信モデルによる作成手順

ピア・ツー・ピア通信モデルよる DirectPlay を用いたアプリケーションの作成手順を図示し,以降でこの作成手順に従って作成したアプリケーションの処理の流れと主な部分の説明を行うことにする.

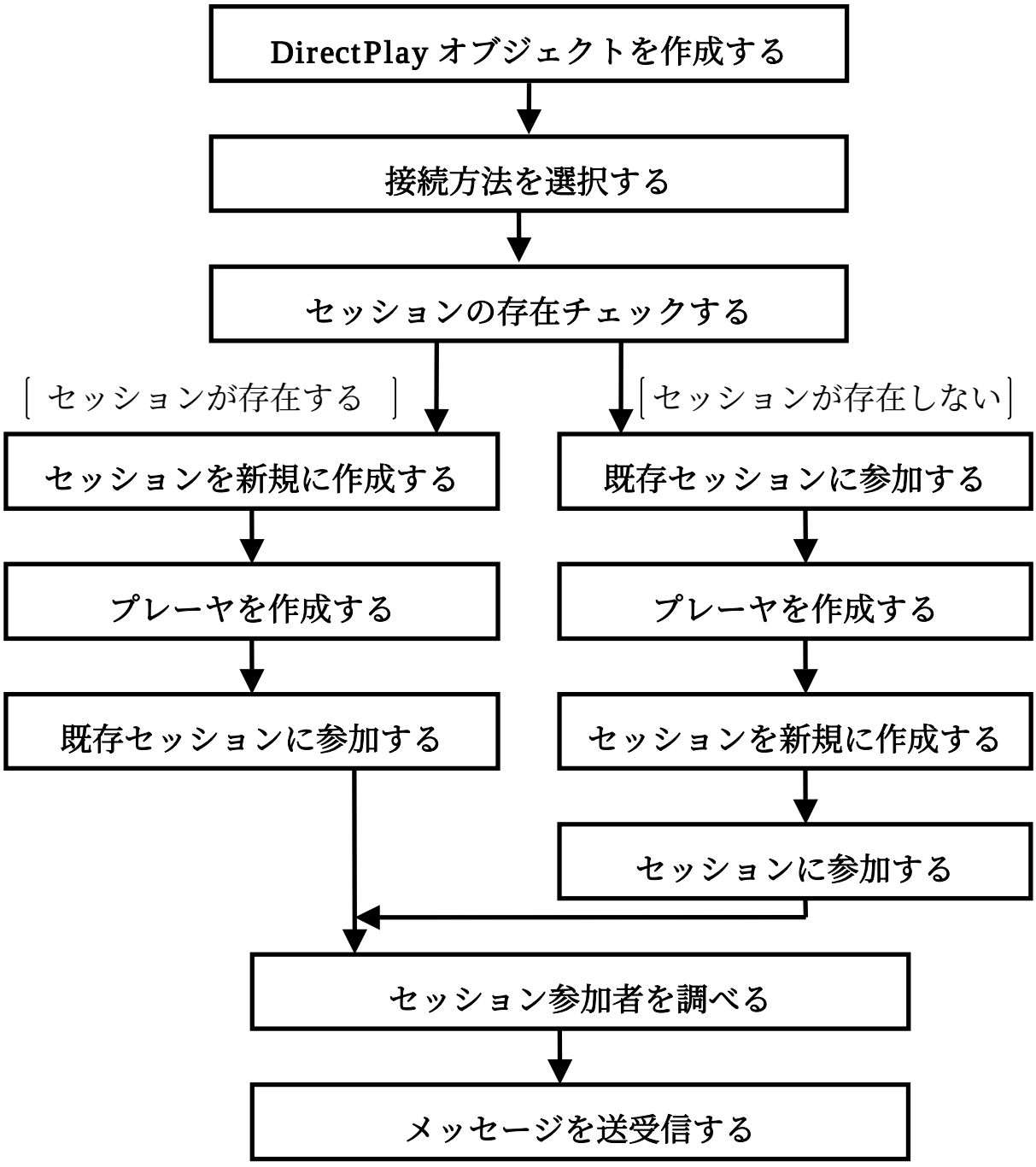

図 4.1　　DirectPlay アプリケーション作成手順

4.1 作成手順と主に使用したオブジェクトの説明

ここではピア・ツー・ピア通信モデルを用いてセッションに参加するまでのアプリケーションの処理内容と,使用したオブジェクトの主なものについての説明を行う.

4.1.1 接続方法を選択する

プログラムを実行した時に接続を初期化するために呼び出されるフォーム,このフォーム経由で行われる処理,およびその処理を行う上で使用されるオブジェクトについて説明する.

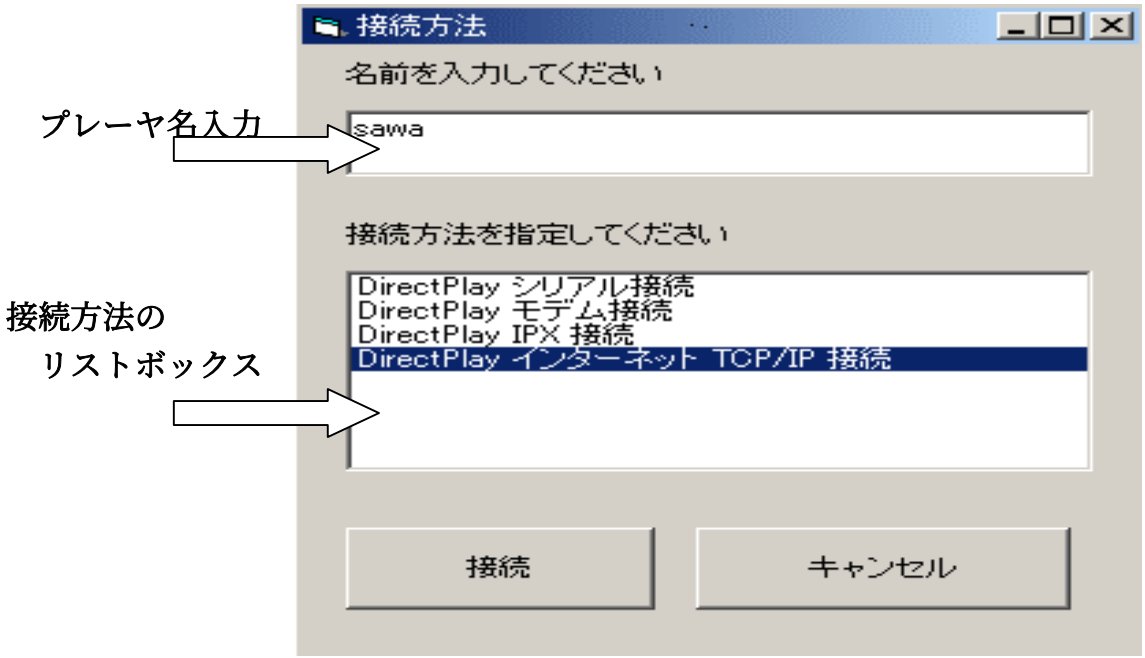

図 4.2. 接続設定を行うフォーム画面

このフォームは,サービスプロバイダとの接続を初期化することを目的としている.提示された接続方法の中から一つを選び,プレーヤ名を入力し,接続ボタンをクリックすることで接続が初期化されるようになっている.

サービスプロバイダとの接続を初期化するためには DirectPlay オブジェクトが必要である.そのため DirectPlay オブジェクトと EnumConnections オブジェクトを取得する.そして接続可能または不可能に限らず全ての接続方法の一覧を取得する.接続ボタンがクリックされた時 DirectPlayAddress オブジェクトを使

用することで一覧から選択された接続方法が取得され,サービスプロバイダとの接続の初期化が完了する.

4.1.2 セッションの場所

参加したいセッションが存在するユーザ対して,セッションの場所を指定できるようにしておく必要がある.ここでは TCP/IP 接続を選択した場合に呼び出されるダイアログ,このダイアログで行われる処理およびその処理を行うために使用するオブジェクトについて説明する.

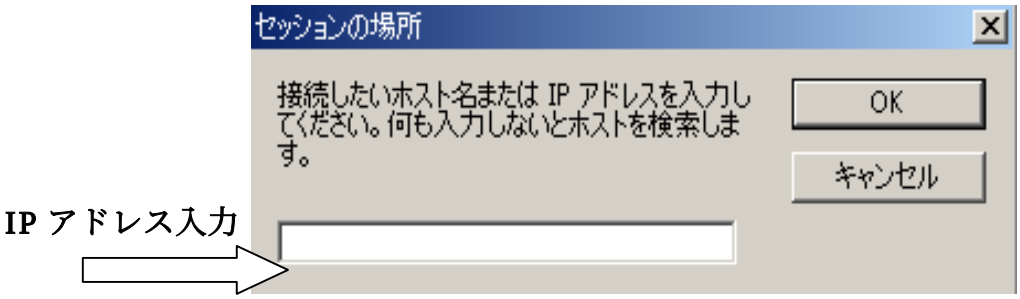

図 4.3　セッションの場所のダイアログ画面

上に示した図は TCP/IP 接続を選択した場合のダイアログである.接続したい相手がいる場合にはその相手の IP アドレスを入力する.接続したい相手がいない場合には何も入力しないで OK ボタンを押す.この場合には全てのセッションを検索する.

4.1.3 セッションの選択と参加

接続したいセッションが無いユーザが新たに検索されたセッションを選択したり,新規にセッションを作成するためには,セッション名や接続したいセッション名が必要となってくる.ここではセッションの選択および参加を行うフォーム,このフォームを用いて行われる処理およびその処理で使用されるオブジェクトについて説明する.

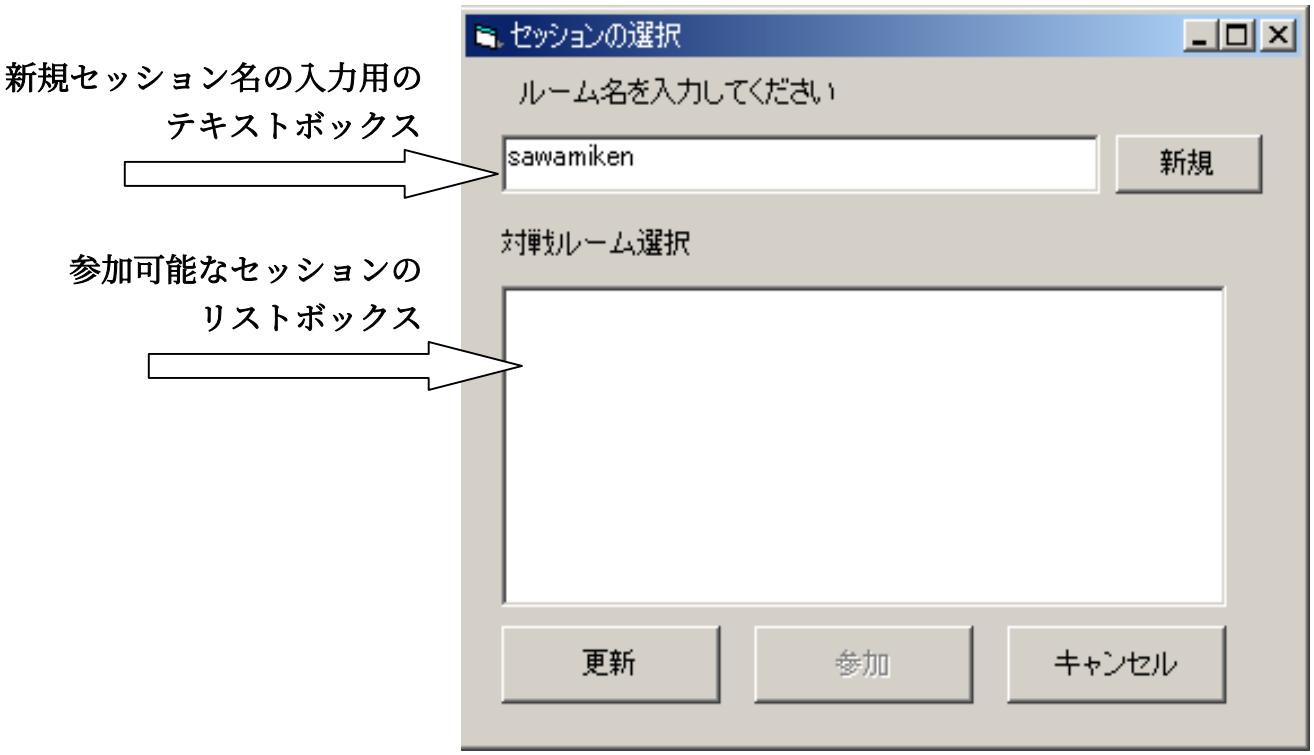

図 4.4　セッションの選択および参加を行うフォーム画面

上記のフォームでは,参加可能なセッションがリストボックスに出力される.既存セッションに参加する場合にはリストボックスに表示されたセッションを選び,参加ボタンをクリックする.新たにセッションを作成する場合にはセッション名を入力し新規ボタンをクリックする.

セッションを作成する上で使用した関数は DirectPlayEnumSessions オブジェクトであり,これを使用してセッションの情報を取得する.セッションに参加する場合には DirectPlay4. Open メドッドを使用して選択したセッションに参加する.更新ボタンが押された場合には,セッションを検索して新たに参加可能なセッションがリストに表示されることになる.新規にセッションを作成する場合には, DirectPlay.CreateSessionData メソッドを用いて,セッションを作成す

る上で必要なデータを取得する.

4.1.4 メッセージの送受信

参加したセッションの参加者を表示したり,ゲームやチャットメッセージを送受信するためには,セッションの参加者リストやチャットの送信メッセージを入力するテキストボックスや,受信したチャットメッセージを表示するテキストボックスが必要となる.ここではメッセージの送受信とゲームを行うフォーム,このフォームを用いて行われるチャットの処理およびその処理で使用されるオブジェクトについて説明する.

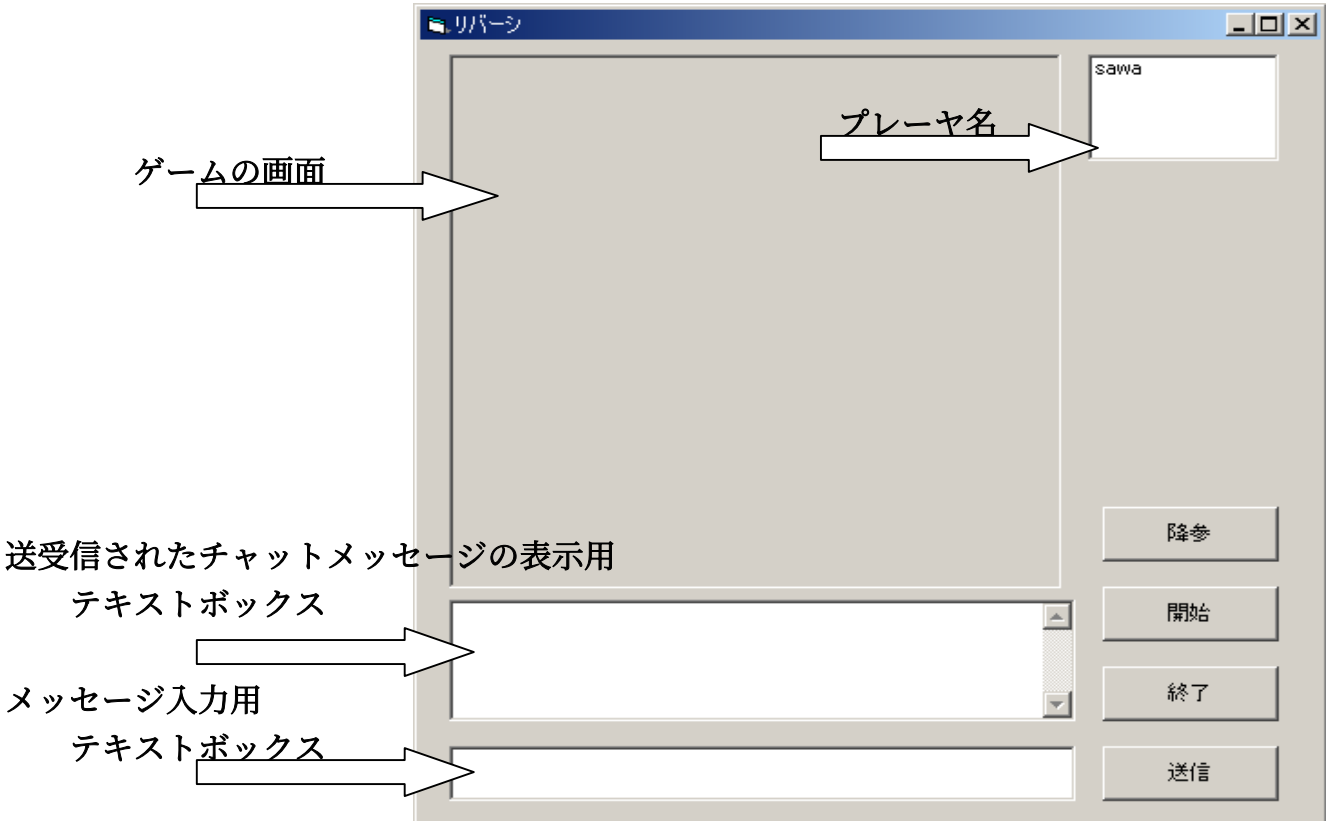

図 4.5　チャットとゲームを行うフォーム画面

上記のフォームによりチャットおよびゲーム画面の表示を行う.メッセージ入力用テキストボックスにメッセージを入力する.送受信されたチャットメッセージ表示用テキストボックスには送受信したチャットメッセージの両方をログのようにして出力する. そこで送信されたチャットメッセージの表示用テキストボックスに出力された文字の長さが 10000 バイトを超えたら,送受信されたメッセージの表示用テキストボックスに格納されているデータの終わりから 9000 バイト残し,古いログを削除するようにした.

チャットを行うえるようにするため,DirectPlay4 Send メソッドを使用して

メッセージの送信を行い,DirectPlay4 Receive メソッドを使用してメッセージの受信を行う.

5. ゲーム

私達がオンラインゲームの例としてリバーシと呼ばれているものを作成した.ここではリバーシの対戦がネットワークを介してどのように行われるかを説明していく.

まず,最初に以降のプログラムの説明を簡単にするために,ゲームを行う上で使用されるプロシージャ名とそのプロシージャが行う処理内容を表にして示しておくことにする.またリバーシが行われている最中の frmgame フォーム画面を伴わせて示しておく.

プロシージャ名	行われる処理の内容
check	黒白の駒の置けるマスの数の検索
DirectXExent_DXCallBack	メッセージの受信を監視
cmdkousan_Click	降参した時の処理
kaisi	リバーシ盤の描画
kekka	駒の変化を認識させる
komabyouga	送信されてきた情報をもとに駒を描画
picbase_MouseDown	picbase 上をクリックした時の処理
syouhai	勝敗が決まったかどうか検索

表 5.1 ゲームを行う上でプロシージャ名と行われる処理内容

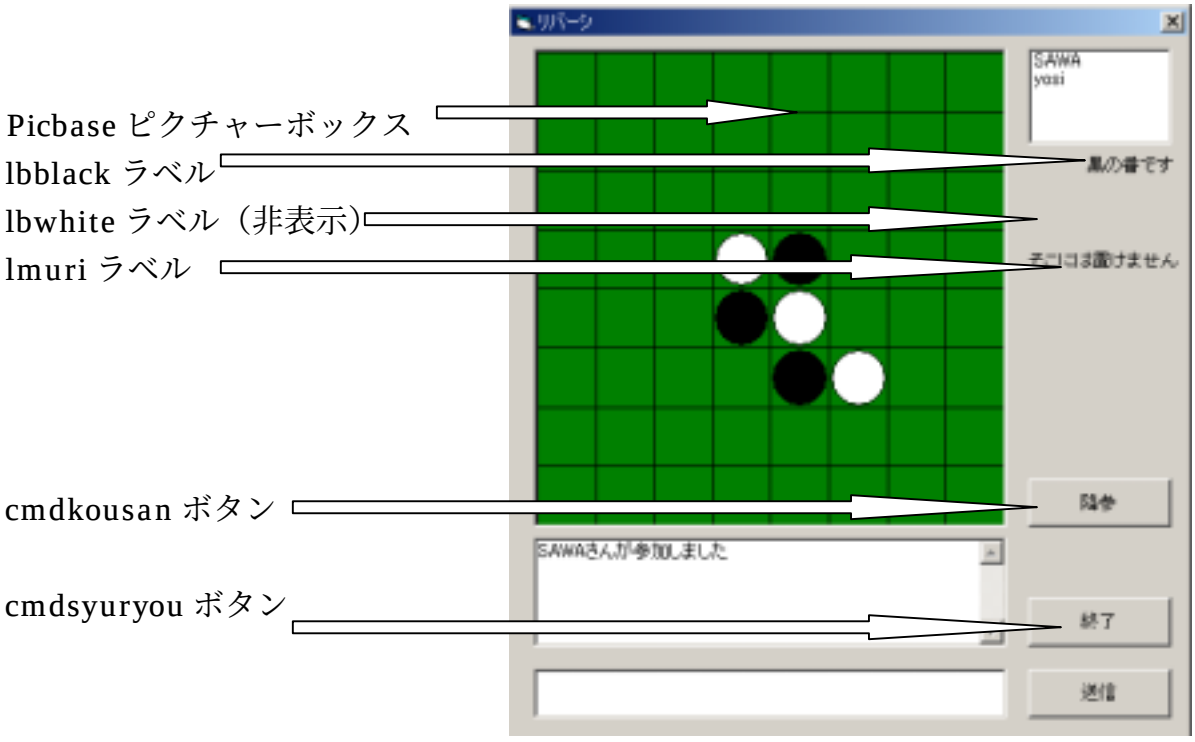

図 5.1　対戦中の frmgame フォーム画面

5.1 リバーシ盤と駒の描画方法

ここでは,リバーシ盤を picbase ピクチャーボックスに表示する方法と,描かれたリバーシ盤に駒を描画するための方法を説明する.

5.1.1 リバーシ盤の線の描画

リバーシ盤は 8×8 の 64 マスで構成されている.従ってピクチャーボックスに描かれるべき線の数は縦線 9 本横線 9 本である.合計 18 本の線を描画する為に picbase ピクチャーボックスの scaleheight を 8 で割った値が用いられる.その値をwとすると次のプログラムを用いることで盤面の線を描いている.

```
For i = 0 To 8
  picbase.Line (i * w, 0)-(i * w, 8 * w), QBColor(0)  ・・・(1)
  picbase.Line (0, i * w)-(8 * w, i * w), QBColor(0)  ・・・(2)
Next i
```

図 5.2　マス目を構成する線を描くためのプログラム

線を描くオブジェクト名．Line(描く線の始点の x 座標，描く線の始点の y 座表)－(描く線の終点の x 座標，描く線の終点の y 座標)，色の指定　という形式で描画のためのプログラムを書く.上記プログラム(1)で picbase オブジェクトに黒の横線を引いている.プログラム(2)で picbase オブジェクトに黒の縦線を引いている.

5.1.2 駒の描画

描かれた盤面に駒を描く場合には,リバーシ盤の線を描く為に用いたwと w を 1/2 倍した値が用いられる.wを 1/２倍にした値を dw とすると,次のプログラムを用いることで駒を描画する.

```
picbase.FillStyle = 0
 picbase.FillColor = QBColor(15)
 picbase.Circle (3 * w + dw, 3 * w + dw), 0.9 * dw  ・・・(1)
```

図 5.3　盤面に白駒を描くプログラム

円を描くオブジェクト名．Circle(描く円の中心ｘ座標, 描く円の中心ｙ座標), 半径という形式で描画のためのプログラムを書く.プログラム(1)は,picbase オブジェクトに中心座標(3w＋dw，3w＋dw)で半径 0.9dw の円を描くということを

意味する.リバーシの駒のため描いた円の中を塗りつぶすために fillstyle プロパティと fillcolor プロパティを用いる.

5.2 ゲーム開始方法

セッションを新規に作成したプレーヤ（ホストプレーヤ）が先手(黒)となるようにしている.そして,ホストプレーヤにはプレーヤを作成する際に必要となるデータ formal name に”playerblack”と設定する.既存のセッションに参加したプレーヤが後手(白)になるようにするため,セッションに参加したプレーヤの formal name を”playerwhite”と設定する.

```
Private Sub cmdsinki_Click()
                .
                .
                .
gmyplayername=
frmsetuzoku.txtname.Text
playerhandle = "playerblack"
                .
                .
                .
End Sub
```

```
Private Sub cmdsanka_Click()
                .
                .
                .
gmyplayername
=frmsetuzoku.txtname.Text
playerhandle = "playerwhite"
                .
                .
                .
End Sub '
```

図 5.3 frmsession の cmdsanka_Click と cmdsinki_Click の一部

ホストプレーヤのゲーム画面は,新規のセッションに新たなプレーヤが参加してくるまで表示されないことを注記しておく.セッションの参加者が 2 人になり,プレーヤリストが更新された時点で,ホストプレーヤのゲーム画面が kaisi プロシージャによって描画される.セッションに参加したプレーヤ側の画面は frmgame フォームが呼び出された時点で kaisi プロシージャによって描画されることになる.

5.3 駒が置けるかどうかの判定

マウスにより picbase ピクチャーボックス上をクリックすることによって駒を置いていく.ここでリバーシ盤のマスをクリックしたという事象をどうように認識させるかが問題となる.クリックした位置の座標(X,Y)を,線を描くために利用した変数 w で割ることにより求めた値を用いて,ピクチャーボックス内の駒の配置を配列 f(9,9)の要素により表すことができる.値 0 と 9 は壁であることを意味することにした.マスの状態としては何も置いてなければ配列の要素の値を 0 とし,黒の駒が置いてあれば１,白の駒が置いてあれば 2 として配列 f に格納する.例えば,右から 3 番目で左から 3 番目のマスに黒の駒が有れば,f(3,3)=2 となる.図 5.4 にマウスによりピクチャーボックスがクリックされた時の処理の流れを示し,以下に各処理の内容を説明する.

分岐 1:クリックした地点に駒が有るのか無いのかを判断する.駒が無い場合は 2 の分岐へ移動,駒が有る場合にはプロシージャを終了する.

分岐 2：順番を表す変数 c の値によって場合分けする.ｃの値が 1 の場合とｃの値が 0 の場合とそれ以外の場合です.ｃ=1 なら黒の番であり,ｃ=0 なら白の番である.そのため,図 5.5 には黒の番の場合の処理を詳しく図示し,白の番の場合の流れ図は省略した.順番が黒の番の場合,３の分岐へ移動する.どちらにも当てはまらない場合はプロシージャを終了する.

分岐 3：クリックしたプレーヤの formal name が先手”playerblack”であるかどうか判断する.先手ならば,ループ 1 へ移動し,そうでなければプロシージャを終了する.

ループ 1：配列要素を変化させるための変数である dx を－1 から１までループさせる.

ループ 2：配列要素を変化させるための変数である dy を－1 から 1 までループさせる.

分岐 4：クリックしたマスの隣 8 方向のいずれかが白の駒であり,かつリバーシ盤の中であれば,2 の処理に移動し,そうでなければループカウンタが１増える.

処理 2：配列の要素変数を隣の白の駒の配列の要素にする.kx,ky

ループ 3：隣のマスが白の駒でなくなるまで配列の要素変数をループカウンタ分だけ加算する.

分岐 5：隣のマスが黒の駒かどうかを判断する.黒の駒ならループ 4 に移動する.

ループ 4：kx＝rx,ky＝ry となるまで処理 4 が行われる.

処理 4：加算されたループカウンタを 1 ずつ減らしながら, kx＝rx,ky＝ry となるまで黒の駒を描画していく.

分岐 6：クリックされたマスに黒の駒があるかどうか判断する.

処理 5：黒の駒があれば,白の駒の置く場所があるかどうかチェックし,さらに勝敗が決まっていないかを判断する.

分岐 7：白の駒の置く場所が無いかどうか判断する.白の駒の置く場所が無ければ処理 6 へ移動し,白の駒の置く場所が有れば処理 7 へ移動する.

処理 6：lmuri ラベルに”白はパスです”と表示し,置かれた駒のデータを対戦相手のプレーヤに送信する.

処理 7：順番を表す変数 c を 1 減らし,置いた駒のデータを対戦相手のプレーヤに送信する.

分岐 8：クリックしたマスに駒が置けない場合.

処理 8：lmuri ラベルに”そこには置けません”と表示する.

以上述べてきた一連の処理を行うことで,マウスでクリックした時にリバーシのルールに沿った表示画面を得ることができる.

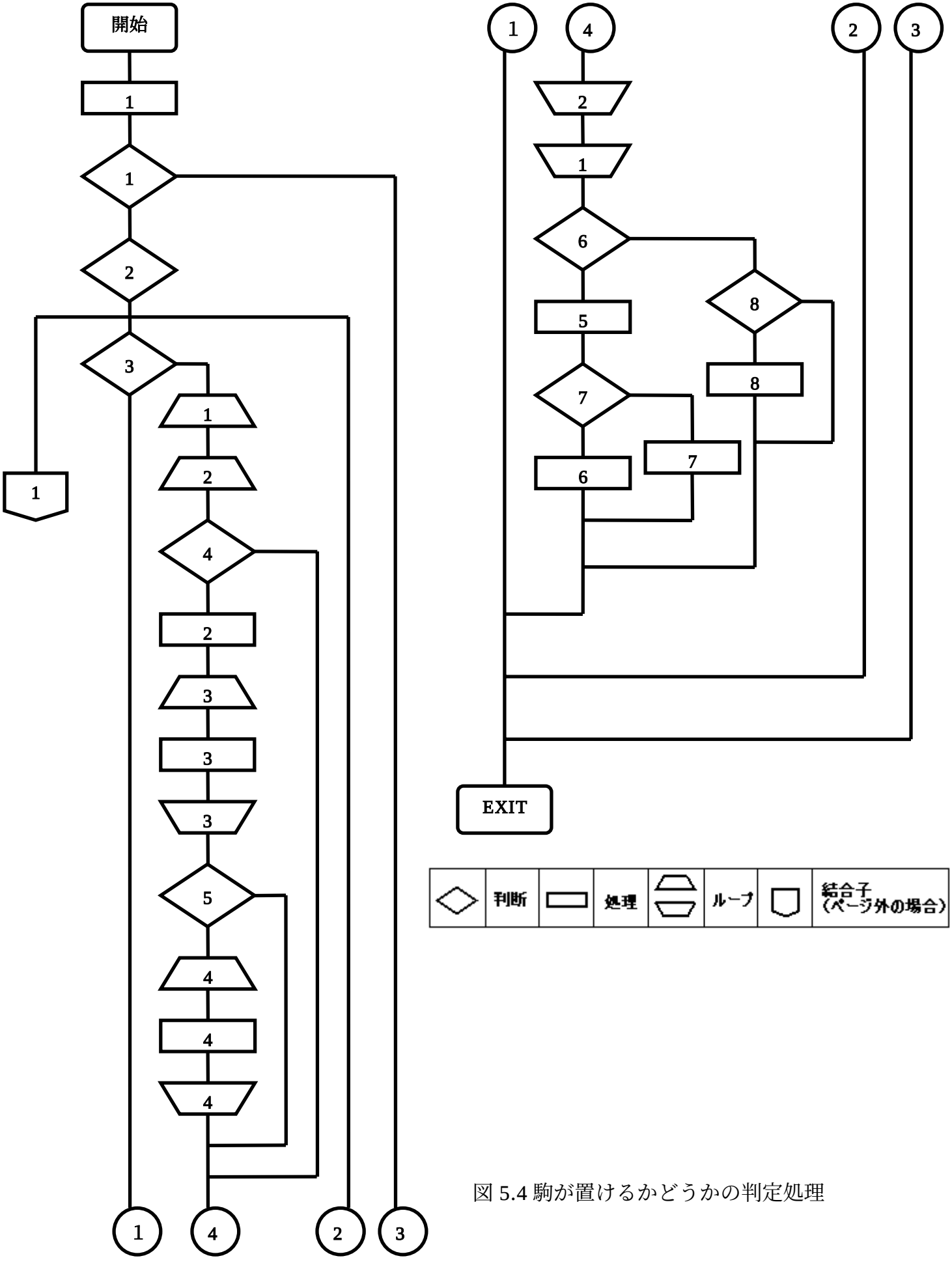

図 5.4 駒が置けるかどうかの判定処理

5.3 に示した処理番号参照

6. メッセージ(ゲームデータ)の送受信

メッセージの送受信には DirectPlay4.Send メソッドと DirectPlay4.Receive メソッドを用ている.メッセージにはゲームメッセージとチャットメッセージがある.両者を区別するために module1 では MSG_TYPES という名前で定数の集合を定義している.メッセージを作成する際には先頭に予め定義された定数名を格納して送ることにする.メッセージを受信した場合,それがシステムメッセージかどうかを区別する.システムメッセージの場合にはプレーヤの入退室を知らせるメッセージを送受信されたメッセージの表示用テキストボックスに表示する(図 4.5).システムメッセージではない場合には,MSG_TYPES を調べ受信したデータを基にした処理を行う.表 6.1 に MSG_TYPES の種類を示す.

MSG_CHAT	チャットメッセージ
MSG_KOMA	駒のデータ
MSG_BLACKKOUSAN	黒が降参を希望している
MSG_WHITEKOUSAN	白が降参を希望している

表 6.1　MSG_TYPES の種類

6.1　メッセージ送信方法

メッセージを送信するための方法を,置かれた駒の色と場所のデータを送信するプログラムを例にして説明する.以下に,置かれた駒のデータを送信するためのプログラムとこれに対応した説明を示すことにする.

```
Dim komamsg As DirectPlayMessage                          ・・・(1)
Set komamsg = gobjdplay.CreateMessage                     ・・・(2)
                    ・
                    ・
                    ・
Call komamsg.WriteLong(MSG_KOMA)                          ・・・(3)
Call komamsg.WriteShort(ry)                               ・・・(4)
Call komamsg.WriteShort(rx)                               ・・・(5)
Call komamsg.WriteShort(1)                                ・・・(6)
Call gobjdplay.Send(gmyplayerid, DPID_ALLPLAYERS, _
DPSEND_GUARANTEED, komamsg)                               ・・・(7)
```

図 6.1　置かれた駒のデータを送るプログラム

(1) DirectPLayMessage オブジェクトを初期化する.
(2) メッセージオブジェクトを作成する.
(3) 送信メッセージに駒のデータを表す MSG_KOMA という定数を格納する.
(4) 送信メッセージに置かれた駒の配列要素変数 rx を格納する.
(5) 送信メッセージに置かれた駒の配列要素変数 ry を格納する.
(6) 送信メッセージに駒の色が黒であることを表す定数 1 を格納する.
(7) 送信メッセージ komamsg を,保証つきのメッセージとして全てのプレーヤに送信する.

6.2　メッセージ受信方法

メッセージの受信は,DirectXExent_DXCallBack プロシージャによって監視する.さらに gobjdplay.GetMessageCount メソッドにより受信したメッセージが存在するかをチェックする.もし受信メッセージがあれば,DirectPlay4.Receive メソッドでメッセージキューからメッセージを取り出す.取り出したメッセージの最初の倍長整数値を調べ,その値によって以降の処理をさせる.倍長整数値としては MSG_TYPES の中のいずれかとなる.次にメッセージを受信するプログラムの一部を示す.

```
Select Case msgtype
    Case MSG_CHAT
            ・
            ・
    Case MSG_KOMA
            ・
            ・
            ・
    Case MSG_BLACKKOUSAN
            ・
            ・
            ・
    Case MSG_WHITEKOUSAN
            ・
            ・
            ・
End Select
```

図 6.2　受信したメッセージを MSG_TYPES によって場合分けする例

6.3　対戦相手が入力した駒の描画

対戦相手が変化させた駒の描画が行われるまでの処理の流れを説明する.例えばプレーヤ A(先手)とプレーヤ B(後手)がリバーシを行っているとする.次に示す図 6.3 の矢印で示されるマスにプレーヤ A が黒駒を置いたとする.すると picbase_MouseDown プロシージャによりプレーヤ A の picbase ピクチャーボックス上で駒を置く描画が行われ,リバーシのルールに沿った形で駒の色が変化する.そして,プレーヤ A が置いた駒の色と駒を置いた位置情報がプレーヤＢに送られる.

情報を送信する方法については 6.1 のメッセージ送信方法に示した通りである.送信される情報としては,駒が置かれたマスに相当する 2 次元配列ｆ(5,6)の配列要素の指標 5,6 と,黒駒を表す数字１である.情報を受信したプレーヤ B のプログラムでは,受信したメッセージを解読していく.解読の流れとしては,MSG_KOMA という定数が読み取られ,駒のデータだと判断される.そして 2 次元配列要素の指標と黒駒を表す数字 1 が取り出され,取り出されたメッセージを引数として komabyouga プロシージャに渡し,プレーヤ B の picbase ピクチャーボックス上の駒をプレーヤ A の picbase ピクチャーボックスで生じた変化と同じようにして変化させる.komabyouga プロシージャについては付録プログラムを参照して欲しい.

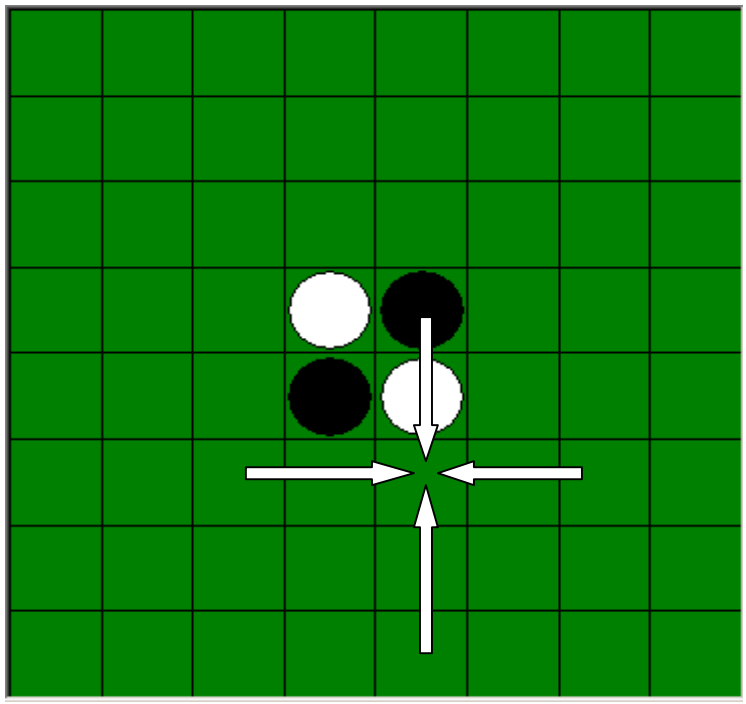

図 6.3　駒を置く位置

７．まとめ

今回,我々は DirectX を使用したオンラインゲームの例としてリバーシのオンラインゲームを作成した.その結果,DirectPlay を使用することによって,どのようなネットワークを使ってマシン同士を接続するかについてそれ程考えなくても良いことが分かった.それらのことからプログラマは,通信部分が完成した時点で,それ以降はゲーム内容のみに専念することができると考えられる.そして,通信部分はどのようなオンラインゲームであっても共通の部分が多いことから,インターネット接続した定形の通信処理が使える環境では,ネットワークを用いた別のゲーム作成を行う場合,大部分の作業はゲーム本体の作成作業になることが分った.

今後オンラインゲーム作成に関する敷居はますます低くなりDirectXを用いた素晴らしいオンラインゲームが数多く作成されるものと考える.
オンラインゲームを作成する際に,私たちの研究が役立てばと思います.

8. 今後の課題

今後の改良点としてはロビー・サーバを利用し,1 台のマシンに対する負荷を少なくし,IP アドレスを利用せず,簡単に対戦相手を探すことができるシステムを作成すること,ゲームプログラムの完成度を上げていくことである.

9．参考文献

[1]Visual Basic ネットワークゲーム プログラミング David Allen 著 相良毅訳 ソフトバンク株式会社 1997 年
[2]基礎からの TCP/IP ネットワーク実験プログラミング　村山公保　オーム社 2001 年
[3]Ver.6.0Visual Basic 初級プログラミング入門[上][下]　河西朝雄　技術評論社 1999 年

10．参考ホームページ

[1]http://www.microsoft.com/

11. 謝辞

一年間,最後まで至らない私達にご指導賜わった澤見英男先生に深く心から感謝致します.また,2001 年度の澤見研究生,木村研究生と数理情報学科の諸先生方に深く心から感謝致します.有り難う御座いました.

2002 年 3 月 20 日

12.付録プログラム

以下に作成したプログラムを示す.本文中に説明がある場合にはその場所を示すようにしている.本文と併せて見て欲しい.

```
Implements DirectXEvent
Dim flag As Boolean   '特定の相手が選ばれたかどうかを格納する変数
Dim w As Integer      'pibase の幅を 8 で割った商を格納する変数
Dim dw As Integer     'w を 2 で割った商を格納する変数
Dim f(9, 9) As Integer   'リバーシの盤を仮想的に表現する配列変数
Dim c As Integer          '順番を表す変数
Dim i As Integer          'ループカウンタ
Dim j As Integer          'ループカウンタ
Dim bcount As Integer     '黒の置ける場所の数を格納する変数
Dim wcount As Integer     '白の置ける場所の数を格納する変数

‘リバーシ盤の描画と最初に配置される駒 4 つの描画(5.1 参照)
Public Sub kaisi()
ScaleMode = 3
w = Int(picbase.ScaleHeight / 8)
dw = w / 2
picbase.BackColor = QBColor(2)
'全ての配列要素を初期化
For i = 0 To 9
   For j = 0 To 9
      f(i, j) = 0
   Next j
Next i
'線を縦横 8 本引く
For i = 0 To 8
   picbase.Line (i * w, 0)-(i * w, 8 * w), QBColor(0)
   picbase.Line (0, i * w)-(8 * w, i * w), QBColor(0)
Next i
'真ん中に黒白の駒を描く
picbase.FillStyle = 0: picbase.FillColor = QBColor(15)
```

```
  picbase.Circle (3 * w + dw, 3 * w + dw), 0.9 * dw
  picbase.Circle (4 * w + dw, 4 * w + dw), 0.9 * dw
picbase.FillStyle = 0: picbase.FillColor = QBColor(0)
  picbase.Circle (3 * w + dw, 4 * w + dw), 0.9 * dw
  picbase.Circle (4 * w + dw, 3 * w + dw), 0.9 * dw
'置かれた白黒の駒を配列に格納
f(4, 4) = 2
f(5, 5) = 2
f(4, 5) = 1
f(5, 4) = 1
lbblack.Visible = True
cmdkousan.Visible = True
cmdkousan.Enabled = True

c = 1

End Sub

'送信ボタンがクリックされた時の処理
Private Sub cmdsousin_Click()
If txtmsg = "" Then Exit Sub

Dim message As DirectPlayMessage
Set message = gobjdplay.CreateMessage
Call message.WriteLong(MSG_CHAT)
Call message.WriteString(txtmsg)
If flag = True Then
   Call gobjdplay.Send(gmyplayerid, lstplayerlist.ItemData(lstplayerlist.ListIndex),
DPSEND_GUARANTEED, message)
   Call tuikalog(gmyplayername & "to" & lstplayerlist.List(lstplayerlist.ListIndex) & ">" &
txtmsg)

Else
   Call gobjdplay.Send(gmyplayerid, DPID_ALLPLAYERS, DPSEND_GUARANTEED,
message)
   Call tuikalog(gmyplayername & ">" & txtmsg)
```

```
End If
'次のメッセージ用に入力ボックスをセットアップ
txtmsg = ""
txtmsg.SetFocus

End Sub

'終了ボタンがクリックされた時の処理
Private Sub cmdsyuryou_Click()
Unload Me
Unload frmsessions
Unload frmsetuzoku
kaihoudplay
End Sub

‘Form が Load された時の処理
Private Sub Form_Load()
kousinplayerlist
txtmsg = ""
txtlog = ""
End Sub

‘プレーヤリストを更新する処理
Public Sub kousinplayerlist()
Dim enumplayers As DirectPlayEnumPlayers
Dim i As Integer

On Local Error GoTo failed
lstplayerlist.Clear
Set enumplayers = gobjdplay.GetDPEnumPlayers("", DPENUMPLAYERS_ALL)
For i = 1 To enumplayers.GetCount
   Call lstplayerlist.AddItem(enumplayers.GetShortName(i), i - 1)
   lstplayerlist.ItemData(i - 1) = enumplayers.GetDPID(i)
Next i

failed:
```

```
MsgBox ("プレーヤリストの更新に失敗しました")
End Sub

‘プレーヤリストがクリックされた時の処理
Private Sub lstplayerlist_Click()
Dim i As Integer
i = lstplayerlist.ListIndex
If (i > 0) And (lstplayerlist.ItemData(i) <> gmyplayerid) Then
 cmdsousin.Enabled = True
 flag = True
Else
 cmdsousin.Enabled = False
 flag = False
End If
End Sub

プレーヤリスト上でキーが押された時の処理
Public Sub lstplayerlist_KeyPress(KeyAscii As Integer)
lstplayerlist_Click
End Sub

‘ログを追加する処理
Public Sub tuikalog(newtext As String)
txtlog.Text = txtlog.Text & newtext & vbCrLf

'古いログを廃棄
If Len(txtlog.Text) > 10000 Then
   txtlog.Text = Right(txtlog.Text, 9000)
End If

txtlog.SelStart = Len(txtlog.Text)

End Sub

‘メッセージの受信の監視と受信したメッセージの解読及び解読したメッセージに対する処理
(6.2 参照)
```

```
Private Sub DirectXEvent_DXCallBack(ByVal eventid As Long)
Dim fromplayerid As Long
Dim fromplayername As String
Dim toplayerid As Long
Dim msgtype As Long
Dim junk As Long
Static dpMsg As DirectPlayMessage
Dim changedplayerid As Long
Dim df(9, 9) As Integer
Dim di As Integer
Dim dj As Integer
Dim dr As Integer


If gobjdplay Is Nothing Then Exit Sub

On Error GoTo msgerror

If gobjdplay.GetMessageCount(gmyplayerid) = 0 Then Exit Sub

Set dpMsg = Nothing
Set dpMsg = gobjdplay.Receive(fromplayerid, toplayerid, DPRECEIVE_ALL)
msgtype = dpMsg.ReadLong()
'システムメッセージかどうかで分岐
If fromplayerid = DPID_SYSMSG Then
   Select Case msgtype
      Case DPSYS_CREATEPLAYERORGROUP, DPSYS_DESTROYPLAYERORGROUP
         junk = dpMsg.ReadLong
         changedplayerid = dpMsg.ReadLong

         If msgtype = DPSYS_CREATEPLAYERORGROUP Then
            Dim name As String
            name = gobjdplay.GetPlayerFriendlyName(changedplayerid)
            Call frmgame.tuikalog(name & "さんが参加しました")
            Call frmgame.kousinplayerlist
            Call frmgame.kaisi
```

```
        Else
          Dim i As Integer
          Dim strname As String

          For i = 0 To frmgame.lstplayerlist.ListCount - 1
            If frmgame.lstplayerlist.ItemData(i) = changedplayerid Then
             strname = frmgame.lstplayerlist.List(i)
             Exit For
            End If
          Next i
          Call frmgame.tuikalog(strname & "さんが退出しました")
        End If

        frmgame.kousinplayerlist
    End Select

Else
    fromplayername = gobjdplay.GetPlayerFriendlyName(fromplayerid)

    Select Case msgtype
        'チャットのメッセージを受け取った場合
        Case MSG_CHAT
            Dim charstr As String

            charstr = fromplayername & ">" & dpMsg.ReadString
            Call frmgame.tuikalog(charstr)
        '駒のデータを受け取った場合
        Case MSG_KOMA

              di = dpMsg.ReadShort
              dj = dpMsg.ReadShort
              dr = dpMsg.ReadShort
              komabyouga di, dj, dr
        '黒が降参を要求した場合
```

```
        Case MSG_BLACKKOUSAN
            MsgBox ("白の勝ちです")
            frmgame.cmdkousan.Enabled = False
        '白が降参を要求した場合
        Case MSG_WHITEKOUSAN
            MsgBox ("黒の勝ちです")
            frmgame.cmdkousan.Enabled = False


    End Select
End If

Exit Sub

msgerror:
MsgBox ("メッセージ読み込みエラー" & Err.Number & " " & Err.Description)
End Sub

‘txtmasg 上でエンターキーを押した時の処理
Private Sub txtmsg_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
    cmdsousin_Click
End If
End Sub

‘降参ボタンを押した時の処理
Private Sub cmdkousan_Click()
Dim lostjunban As String
Dim lostmsg As DirectPlayMessage

Set lostmsg = gobjdplay.CreateMessage

lostjunban = gobjdplay.GetPlayerFormalName(gmyplayerid)

If lostjunban = playerblack Then
  Call lostmsg.WriteLong(MSG_BLICKKOUSAN)
```

```
 Call gobjdplay.Send(gmyplayerid, DPID_ALLPLAYERS, DPSEND_GUARANTEED,
lostmsg)
 MsgBox "白の勝ち", vbExclamation
 cmdkousan.Enabled = False
ElseIf c = 0 Then
 Call lostmsg.WriteLong(MSG_WHITEKOUSAN)
 Call gobjdplay.Send(gmyplayerid, DPID_ALLPLAYERS, DPSEND_GUARANTEED,
lostmsg)
 MsgBox "黒の勝ち", vbExclamation
 cmdkousan.Enabled = False
End If
End Sub

‘結果を求める処理
Private Sub kekka()
Dim black As Integer
Dim white As Integer
For i = 1 To 8
 For j = 1 To 8
  If f(i, j) = 1 Then
   black = black + 1
  ElseIf f(i, j) = 2 Then
   white = white + 1
  End If
 Next j
Next i
End Sub

‘白黒の駒が置ける場所の数を数える処理
Private Sub check()
Dim ex As Integer   'x 座標の差分変数
Dim ey As Integer   'y 座標の差分変数
Dim cx As Integer   '検索用 x 座標変数
Dim cy As Integer   '検索用 y 座標変数
bcount = 0          '白が置く事が出来るマスの数を 0 とする
wcount = 0          '黒が置く事が出来るマスの数を 0 とする
```

```
For i = 1 To 8
 For j = 1 To 8
  If f(i, j) = 0 Then
   cx = i
   cy = j
   For ex = -1 To 1
    For ey = -1 To 1
     If f(i + ex, j + ey) = 1 And (i + ex) >= 1 And (i + ex) <= 8 And (j + ey) >= 1 And (j + ey) <=
8 Then
      cx = i + ex
      cy = j + ey
      Do While f(cx, cy) = 1 And cx >= 1 And cx <= 8 And cy >= 1 And cy <= 8
       cx = cx + ex
       cy = cy + ey
      Loop
      If f(cx, cy) = 2 Then
       wcount = wcount + 1
      End If
     ElseIf f(i + ex, j + ey) = 2 And (i + ex) >= 1 And (i + ex) <= 8 And (j + ey) >= 1 And (j + ey)
<= 8 Then
      cx = i + ex
      cy = j + ey
      Do While f(cx, cy) = 2 And cx >= 1 And cx <= 8 And cy >= 1 And cy <= 8
       cx = cx + ex
       cy = cy + ey
      Loop
      If f(cx, cy) = 1 Then
       bcount = bcount + 1
      End If
     End If
     Next ey
    Next ex
  End If
 Next j
Next i
End Sub
```

```
‘駒が置けるかどうかの判断と変化した駒を描く処理(5.3 参照)
Private Sub picbase_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
‘マウスで picbase.picturebox をクリックした時の処理
Dim komamsg As DirectPlayMessage
Set komamsg = gobjdplay.CreateMessage
Dim junban As String
Dim dk As Integer
'プレーヤの正式名を取得し、junban に保存する
junban = gobjdplay.GetPlayerFormalName(gmyplayerid)

ScaleMode = 3
Dim dx As Integer   'x 座標の差分変数
Dim dy As Integer   'y 座標の差分変数
Dim kx As Integer   '検索用 x 座標変数
Dim ky As Integer   '検索用 y 座標変数

w = Int(picbase.ScaleHeight / 8)
rx = Int((X / w) + 1)
ry = Int((Y / w) + 1)
kx = rx
ky = ry

If f(rx, ry) = 0 Then
 Select Case c
  Case 1
  'マウスをクリックしたプレーヤが黒の駒を置けるプレーヤかどうか判定
  If junban <> "playerblack" Then
   Exit Sub
  Else
   '置こうとしているマスの 8 方向に白の駒があるかどうか探す
   For dx = -1 To 1
    For dy = -1 To 1
     '白の駒があればその隣が白でなくなるまで移動する
     If f(rx + dx, ry + dy) = 2 And (rx + rx) >= 1 And (rx + dx) <= 8 And (ry + dy) >= 1 And (ry
```

```
+ dy) <= 8 Then
          kx = rx + dx
          ky = ry + dy
          Do While f(kx, ky) = 2 And kx >= 1 And kx <= 8 And ky >= 1 And ky <= 8
            kx = kx + dx
            ky = ky + dy
          Loop
          '白の駒でないマスに黒の駒が合った場合
          If f(kx, ky) = 1 Then
          '置こうとしているコマまで差分変数分減らしながら黒の駒を描いていく
            Do Until kx = rx And ky = ry
              kx = kx - dx
              ky = ky - dy
              picbase.FillStyle = 0: picbase.FillColor = QBColor(0)
              picbase.Circle ((kx - 1) * w + dw, (ky - 1) * w + dw), 0.9 * dw
              f(kx, ky) = 1
              kekka
            Loop
          End If
        End If
      Next dy
    Next dx
    If f(rx, ry) = 1 Then
      check
      syouhai
    If wcount = 0 Then
      lmuri.Caption = "白はパスです"
      Call komamsg.WriteLong(MSG_KOMA)
      Call komamsg.WriteShort(rx)
      Call komamsg.WriteShort(ry)
      Call komamsg.WriteShort(1)
      Call  gobjdplay.Send(gmyplayerid,  DPID_ALLPLAYERS,  DPSEND_GUARANTEED,
komamsg)
    Else
      c = c – 1
      Call komamsg.WriteLong(MSG_KOMA)
```

```
      Call komamsg.WriteShort(rx)
      Call komamsg.WriteShort(ry)
      Call komamsg.WriteShort(1)
      Call gobjdplay.Send(gmyplayerid, DPID_ALLPLAYERS, DPSEND_GUARANTEED,
komamsg)

       lmuri.Caption = ""
       lbblack.Visible = False
       lbwhite.Visible = True
     End If
     ElseIf f(rx, ry) = 0 Then
     lmuri.Caption = "そこには置けません"
     End If
   End If
Case 0
   If junban <> "playerwhite" Then
      Exit Sub
   Else
      For dx = -1 To 1
      For dy = -1 To 1
      If f(rx + dx, ry + dy) = 1 And (rx + dx) >= 1 And (rx + dx) <= 8 And (ry + dy) >= 1 And (ry
+ dy) <= 8 Then
         kx = rx + dx
         ky = ry + dy
         Do While f(kx, ky) = 1 And kx >= 1 And kx <= 8 And ky >= 1 And ky <= 8
            kx = kx + dx
            ky = ky + dy
         Loop
         If f(kx, ky) = 2 Then
            Do Until kx = rx And ky = ry
               kx = kx - dx
               ky = ky - dy
               picbase.FillStyle = 0: picbase.FillColor = QBColor(15)
               picbase.Circle ((kx - 1) * w + dw, (ky - 1) * w + dw), 0.9 * dw
               f(kx, ky) = 2
               kekka
```

```
            Loop
           End If
         End If
       Next dy
     Next dx
   If f(rx, ry) = 2 Then
   check
   syouhai
   If bcount = 0 Then
   lmuri.Caption = "黒はパスです"
    Call komamsg.WriteLong(MSG_KOMA)
    Call komamsg.WriteShort(rx)
    Call komamsg.WriteShort(ry)
    Call komamsg.WriteShort(2)
    Call gobjdplay.Send(gmyplayerid, DPID_ALLPLAYERS, DPSEND_GUARANTEED,
komamsg)
   Else
      c = c + 1
      Call komamsg.WriteLong(MSG_KOMA)
      Call komamsg.WriteShort(rx)
      Call komamsg.WriteShort(ry)
      Call komamsg.WriteShort(2)
      Call gobjdplay.Send(gmyplayerid, DPID_ALLPLAYERS, DPSEND_GUARANTEED,
komamsg)

      lmuri.Caption = ""
      lbwhite.Visible = False
      lbblack.Visible = True
   End If
   Else if f(rx,ry)=0 then
    muri.Caption = "そこには置けません"
   End If
   End If
   End Select
End If
End Sub
```

```
‘受信したメッセージが駒のデータの場合受信したデータを基に変化した駒を描画する処理
(6.3 参照)
Public Sub komabyouga(o As Integer, p As Integer, r As Integer)
‘受信した駒のデータを基に色の変わった駒を描画するプロシージャ
Dim ox As Integer
Dim py As Integer
Dim qx As Integer
Dim sy As Integer

If r = 1 Then
   For ox = -1 To 1
    For py = -1 To 1
     If f(o + ox, p + py) = 2 And (o + ox) >= 1 And (o + ox) <= 8 And (p + py) >= 1 And (p + py)
<= 8 Then
        qx = o + ox
        sy = p + py
        Do While f(qx, sy) = 2 And qx >= 1 And qx <= 8 And sy >= 1 And sy <= 8
          qx = qx + ox
          sy = sy + py
         Loop
           If f(qx, sy) = 1 Then
            Do Until qx = o And sy = p
              qx = qx - ox
              sy = sy - py
              picbase.FillStyle = 0: picbase.FillColor = QBColor(0)
              picbase.Circle ((qx - 1) * w + dw, (sy - 1) * w + dw), 0.9 * dw
              f(qx, sy) = 1
              kekka
             Loop
           End If
         End If
        Next py
      Next ox
      check
      syouhai
```

```
      If wcount = 0 Then
        lmuri.Caption = "白はパスです"
      Else
        c = c - 1
        lmuri.Caption = ""
        lbblack.Visible = False
        lbwhite.Visible = True
      End If
ElseIf r = 2 Then
  For ox = -1 To 1
    For py = -1 To 1
     If f(o + ox, p + py) = 1 And (o + ox) >= 1 And (o + ox) <= 8 And (p + py) >= 1 And (p + py)
<= 8 Then
        qx = o + ox
        sy = p + py
        Do While f(qx, sy) = 1 And qx >= 1 And qx <= 8 And sy >= 1 And sy <= 8
          qx = qx + ox
          sy = sy + py
        Loop
        If f(qx, sy) = 2 Then
         Do Until qx = o And sy = p
           qx = qx - ox
           sy = sy - py
           picbase.FillStyle = 0: picbase.FillColor = QBColor(15)
           picbase.Circle ((qx - 1) * w + dw, (sy - 1) * w + dw), 0.9 * dw
           f(qx, sy) = 2
           kekka
         Loop
        End If
      End If
    Next py
  Next ox
  check
  syouhai
  If bcount = 0 Then
   lmuri.Caption = "黒はパスです"
```

```
        Else
         c = c + 1
         lmuri.Caption = ""
         lbwhite.Visible = False
         lbblack.Visible = True
        End If
End If
End Sub

'勝敗を判定する処理
Private Sub syouhai()
Dim black As Integer
Dim white As Integer
For i = 1 To 8
  For j = 1 To 8
      If f(i, j) = 1 Then
          black = black + 1
      ElseIf f(i, j) = 2 Then
          white = white + 1
      End If
    Next j
Next i
check
If black + white = 64 Or black = 0 Or white = 0 Or (bcount = 0 And wcount = 0) Then
      If black > white Then
        MsgBox "黒の勝ち", vbExclamation

      ElseIf black = white Then
        MsgBox "引き分け", vbExclamation

      Else
        MsgBox "白の勝ち", vbExclamation

      End If
  lbblack.Visible = False
  lbwhite.Visible = False
```

```
 lmuri.Caption = ""
 cmdkousan.Enabled = False
 kaisi
End If
End Sub
```