卒業研究報告書

題 目

三次元チェス対戦ゲームの開発

指 導 教 員

石 水　隆 助教

報 告 者

08–1–037–0219

原 田　友 人

近畿大学理工学部情報学科

平成 24 年 1 月 31 日提出

概要

現在の一般家庭で用いられているパーソナルコンピュータの性能は数年前とは比べものにならないほどに上昇している。特に CPU の進化は顕著で計算機能の上昇は当然ながら GPU 機能を内蔵されているものも少なくない。その GPU 内蔵 CPU の出現に伴い多くのアプリケーションで 3D グラフィックが用いられるようになった。近年のごく短い期間で考えた場合でもそれらの内蔵 GPU の性能の上昇は非常に大きく、今後も益々性能が上がっていくであろうことは想像に易い。ならば当然 3D グラフィックを用いたアプリケーションの需要はより高まっていくと思われ、特に 3D ならではの機能を用いたアプリケーションはニーズが高いだろう。そこで本研究では、3D ならではの機能を用いたアプリケーションの開発を行い、開発を通じてユーザにとって使いやすい 3D グラフィックアプリケーションとするにはどうすればいいか研究する。

具体的な内容としては通常のグラフィック表示では表現の難しいものの代表として、3D チェスを作成する。3D チェスは SF 小説などで昔から未来の遊戯を演出する小道具として登場することもあった。今回は 1907 年にドイツのフェルディナント・マック (Ferdinand Maack) によって発明されたラオムシャッハ (Raumschach) と呼ばれる変則チェスゲームの対戦アプリケーションを作成することにした。これは $5 \times 5 \times 5$ の 125 のマスからなる立方体状のチェス盤を利用したもので、通常のグラフィック表示では駒の位置関係が理解しにくく、現実のチェスボードで行うには駒の位置関係の維持等の問題から難しい。このことから 3D グラフィックアプリケーションで表現するのに最適なものであると考えた。

開発にはマイクロソフト社の提供するマルチメディア処理用の API 群である Microsoft DirectX のソフトウェア開発キット、Microsoft DirectX SDK を用い、C++ 言語により Windows アプリケーションとして作成した。

目次

1 **序論** 1

1.1 背景 1

1.2 三次元チェス 1

1.3 本報告書の構成 2

2 **ラオムシャッハ** (Raumschach) 2

2.1 基本的なルール 2

2.2 各駒の動き方 2

2.3 その他のルール 4

3 **アプリケーション** 6

3.1 Windows アプリケーションの基礎 6

3.2 DirectX で使用した機能 11

3.3 ラオムシャッハプログラム 21

3.4 プログラムの仕様 24

4 **結論・今後の課題** 25

謝辞 27

参考文献 28

付録 A Windows **アプリケーションのスケルトンプログラムのソースコード** 29

付録 B Raumschach **アプリケーションのソースコード** 30

B.1 my3dlib.h 30

B.2 my3dlib.cpp 33

B.3 game.cpp 40

B.4 main.cpp 68

1 序論

1.1 背景

旧来よりパーソナルコンピュータが映像を出力するためには、パーソナルコンピュータとグラフィックコントローラ (以下 GPU) 機能を持つビデオカードを接続し、そのビデオカードとディスプレイを接続することで映像を出力するものが主流ではあった。しかし製造コストの低下を求めて文書作成や電子メールの送受信などの描画性能をそれほど必要としないユーザ向けに、CPU やチップセットに GPU 機能が統合されている製品もあった。もちろんこれらの統合 GPU による描画性能は専用のものとして用意されたビデオカードの描画性能と比べると遥かに劣るものである。3D グラフィックなどが用いられたデータを正確に出力できないなどの問題もあった。

だが近年になって低価格帯ビデオカードと同程度の描画性能を持つほどの GPU 内蔵 CPU が一般製品化され販売され始めた [1][2] ことで、世間一般のパーソナルコンピュータの描画性能の底上げといえるような現象が起こった。パーソナルコンピュータの内部の事情などに興味のない人でも、所有するパーソナルコンピュータの描画性能が高いといった具合である。今後もこの流れは変わることがないだろうと予想されるため、描画性能の低いと言われるパーソナルコンピュータでも 3D グラフィックが用いられたコンテンツを手軽に扱うことができるようになるだろうと考えられる。そうなれば 3D グラフィックを用いたコンテンツの需要が高まることは当然のことである。特に旧来の 2D グラフィックで表現するのは難しかった、3D ならではの機能を用いたアプリケーションは間違いなくニーズが高いはずだと思われる。

そこで本研究では 3D ならではの機能を用いたアプリケーションの開発を行い、開発を通じてユーザにとって使いやすい 3D グラフィックアプリケーションとするにはどうすればいいか研究する。

アプリケーションの開発にはマイクロソフト社の提供するマルチメディア処理用の API 群である Microsoft DirectX[5] のソフトウェア開発キット、Microsoft DirectX SDK を用い、C++ 言語により Windows アプリケーションとして作成した。

1.2 三次元チェス

本研究では 3D ならではの機能を用いたアプリケーションとして 3D チェスの対戦ゲームを行う。3D チェスは変則チェスの一種で、通常のチェスが平面上で二次元的に駒を動かすのに対し、駒を三次元的に動かすもののことである。今回はその中でも最も古いものの一つであるラオムシャッハ (Raumschach)[4] と呼ばれる、1907 年にドイツのフェルディナント・マック (Ferdinand Maack) によって発明された 3D チェスのゲームアプリを開発する。

既存の 3D チェスには Three Dimensional Eight Level Chess[7]、3D CHESS[8] 等がある。これらはプレイするには、三次元的に駒を配置する必要があるため、駒の置き方を工夫し、また、盤全体を見やすくなるようにせねばならない。そのため、上記の 3D チェスは実際に次元にチェス盤を設置し駒を動かすよりも、計算機上で動かした方がプレイし易いと思われる。計算機上で 3D チェスをプレイできる既知のソフトウェアとしては Raumschach[9] 等がある。

1.3 本報告書の構成

本報告書では 2 章でラオムシャッハの基本的なルールについて述べる。また、3 章で実際のアプリケーションを作成した際に使用した DirectX の関数の説明、そしてプログラムの実際の流れを説明する。

2 ラオムシャッハ (Raumschach)

本章では本研究でアプリケーションを作成する対象であるラオムシャッハ (Raumschach) について述べる。

チェス (Chess) は世界中で最も広くプレイされているボードゲームである。また Fairy Chess と呼ばれるチェスのバリエーションも多く存在する。ラオムシャッハは Fairy Chess の一種であり、チェスを三次元に拡張したものである。

ラオムシャッハは通常のチェスと同じく、二人のプレイヤーが互いのキングを取り合うゲームである。チェス盤のマスは三次元上に配置され、各駒が移動できる範囲も三次元に拡張されている。

2.1 基本的なルール

ラオムシャッハのチェス盤は立方体を $5 \times 5 \times 5$ に分けた 125 マスの空間で構成される。各階層 (Column) は下層より順に大文字の A〜E で表し、通常のチェス同様列 (File) は小文字の a〜e、行 (Rank) は数字の 1〜5 で表す。各マスは Bb5 のように階層, 列, 行を順に並べて表記される。駒は通常のチェスの 6 種類 (キング (King)・クイーン (Queen)・ビショップ (Bishop)・ナイト (Knight)・ルーク (Rook)・ポーン (Pawn)) に加えて、ラオムシャッハ独自の駒ユニコーン (Unicorn) を合わせ 7 種類であり、白駒は階層 A,B の 1,2 行に、黒駒は階層 D,E の 4,5 列に配置される。表 1 にラオムシャッハで用いられるコマを示す。また、図 1 にラオムシャッハのチェス盤と駒の初期配置を示す。図 1 中の赤い文字は白駒の位置を、黒い文字は黒駒の位置を表す。

ゲームは白と黒に分かれた二人のプレイヤーにより行われる。先手は白でプレイヤーは交互に自分の駒を一回ずつ動かす。パスをすることはできない。自分の手番の時に自分の駒の動ける範囲に敵の駒が存在するならば、敵の駒をチェスボードから取り除き自分の駒を敵の駒のあった場所に移動することができる。(例外:ポーン P) 以下、これを攻撃と記述する相手のキングを攻撃できる状態にする手を「チェック (Check)」という。いかなる場合においてもキングはチェックされる位置に移動することはできない。キングが絶対に逃げられないように追い詰めたチェックを「チェックメイト (Checkmate)」と呼ぶ。双方は相手のキングをチェックメイトすることを目指す。

2.2 各駒の動き方

ラオムシャッハの駒の動き方は、通常のチェスの駒の動きを三次元方向に拡張したものとなっている。以下では (x, y, z) $(\mathrm{a} \leq \mathrm{x} \leq \mathrm{e}, 1 \leq \mathrm{y} \leq 5, \mathrm{A} \leq \mathrm{z} \leq \mathrm{E})$ をラオムシャッハのチェス盤の (列, 行, 階層) 座標とする。

- キング
  キングは全ての方向に 1 マス移動できる。つまり、キングが (x, y, z) にいる場合、$(x \pm 1, y \pm 1, z \pm 1)$, $(x, y \pm 1, z \pm 1)$, $(x \pm 1, y, z \pm 1)$, $(x, y, z \pm 1)$, $(x, y \pm 1, z)$, $(x \pm 1, y, z)$ のいずれか一ヶ所に移動できる。

表 1　ラオムシャッハの駒

略称	駒	個数
K	キング (King)	1
Q	クイーン (Queen)	1
N	ナイト (Knight)	2
B	ビショップ (Bishop)	2
R	ルーク (Rook)	2
U	ユニコーン (Unicorn)	2
P	ポーン (Pawn)	10

ナイト、ビショップ、ルークは xy 平面、yz 平面、zx 平面の全てに対し従来のチェスと同様の動きをする。

- ビショップ
  ビショップを立方体の中心部に置いた場合、各辺の中央に向かう 12 方向に移動できる。すなわち、ビショップは初期位置からベクトル $(0, \pm1, \pm1)$, $(\pm1, 0, \pm1)$, $(\pm1, \pm1, 0)$ のいずれか 1 つの方向へ他の駒または盤端に当たるまで移動できる。
- ルーク
  ルークを立方体の中心部に置いた場合、各面の中央に向かう 6 方向に移動できる。すなわち、ルークは初期位置からベクトル $(0, 0, \pm1)$, $(0, \pm1, 0)$, $(\pm1, 0, 0)$ のいずれか 1 つの方向へ他の駒または盤端に当たるまで移動できる。
- ナイト
  ナイトは各面に向って 2 マス進み、そこから 90 度曲がった方向 (上下左右 4 方向) に 1 マス進んだ計 24 箇所に移動できる。つまり初期位置 (x, y, z) にいるナイトは $(x \pm 2, y \pm 1, z)$, $(z \pm 2, y, z \pm 1)$, $(x \pm 1, y \pm 2, z)$, $(z \pm 1, y, z \pm 2)$, $(x, y \pm 2, z \pm 1)$ のいずれか一ヶ所に移動できる。
- ユニコーン
  ラオムシャッハ独自のユニコーンは立方体の頂点方向八方向に無限大に動くことができる。すなわち、ユニコーンは初期位置からベクトル $(\pm1, \pm1, \pm1)$ のいずれか 1 つの方向へ他の駒または盤端に当たるまで移動できる。
- クイーン
  クイーンはビショップ、ルーク、ユニコーンの全ての動きを兼ね揃えた駒である。すなわち、クイーンは 27 個のベクトル (x, y, z) $(x, y, z = \{-1, 0, 1\})$ のうち 0 ベクトル $(0, 0, 0)$ を除く 26 個のベクトルのいずれか　つの方向へ他の駒または盤端に当たるまで移動できる。

図 2、図 3 に各駒の動き方を示す。図中の×は各駒が移動できるマスであり、そこに敵駒がいればその駒を攻撃することができる。なお、ナイト以外の駒は他の駒を飛び越すことができない。

- ポーン
  ポーンは移動できるマスと攻撃できるマスが異なる。移動する場合、ポーンは前へ 1 マス、もしくは相手陣地に進む方向へ階層を 1 層移動することができる。ポーンの攻撃できるマスは 1 マス進んだ左右のマス、もしくは 1 層進んだ左右と前のマスである。ポーンが攻撃した場合も他の駒の攻撃と同様に敵の

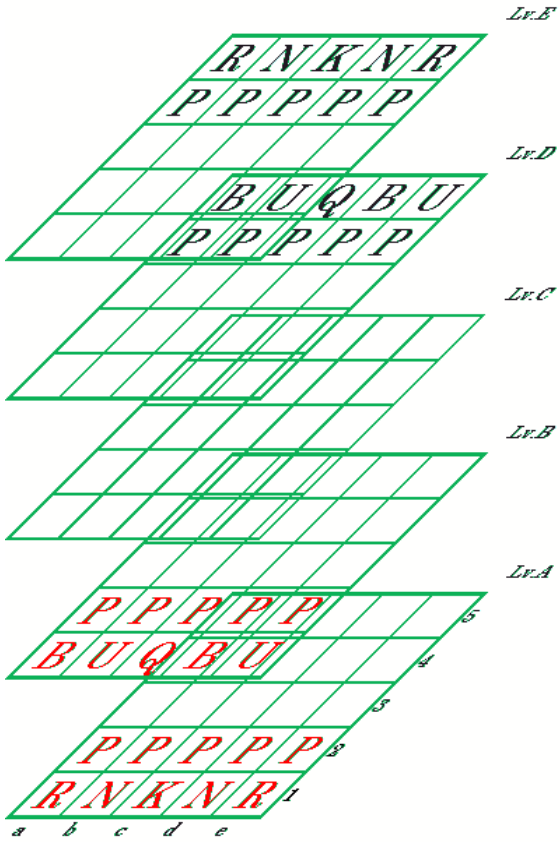

図 1　ラオムシャッハのチェス盤と駒の初期配置

駒のあった場所にポーンを移動させる。つまり初期位置 (x, y, z) にいる白駒のポーンは $(x, y+1, z)$, $(x, y, z+1)$ のどちらかに移動でき、$(x \pm 1, y+1, z)$, $(x \pm 1, y, z+1)$, $(x, y+1, z+1)$ のいずれかに黒駒がいればそれを取ってその位置へ移動できる。また、黒駒のポーンは $(x, y-1, z)$, $(x, y, z-1)$ のどちらかに移動でき、$(x \pm 1, y-1, z)$, $(x \pm 1, y, z-1)$, $(x, y-1, z-1)$ のいずれかに白駒がいればそれを取ってその位置へ移動できる。例えば、白駒のポーンが Cc3 にある場合、Cc4 および Dc3 に進め、Cb4,Cd4,Db3,Dd3,Dc4 に黒駒があればそれを取ってその位置に進むことができる。同様に、黒駒のポーンが Cc3 にある場合、Cc2 および Bc3 に進め、Cb2,Cd2,Bb3,Bd3,Bc2 の白駒を攻撃することができる。

図 4 にポーンの動き方を示す。図中の×はポーンが移動できるマスであり、○は敵駒がいる場合、それを攻撃できるマスである。また、通常のチェスでは、初期配置から動いていないポーンは 2 マス前進する 2 段跳ね (2-step initial move) を行えるが、ラオムシャッハでは 2 段跳ねは行えない。

2.3　その他のルール

その他のルールとして、ラオムシャッハにはステールメートおよびプロモーションがある。

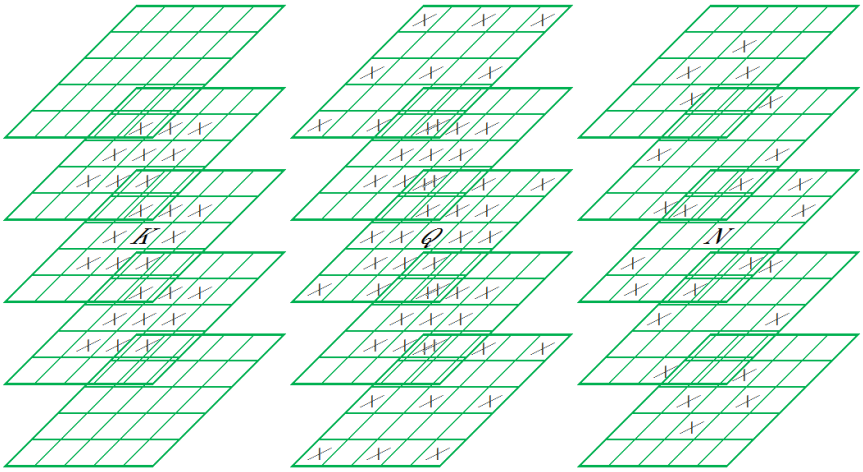

図 2　キング・クイーン・ナイトの動き方

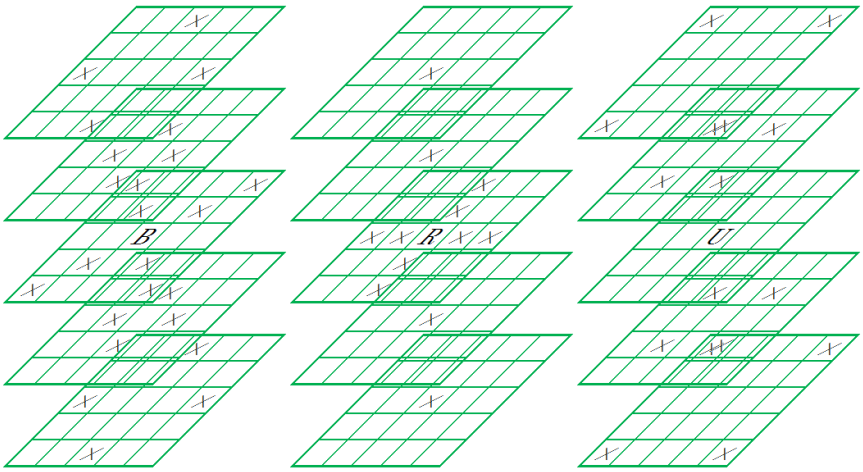

図 3　ビショップ・ルーク・ユニコーンの動き方

- ステールメイト
  先に記述したとおりキングはチェックされる位置に移動することはできない。よって残りの駒が極端に少なくなった場合、どの駒も動かすことのできない状態が発生することがある。これをステールメイト (Stalemate) といい、この時ゲームは引き分けとなる。
- プロモーション
  ポーンがそれ以上進めない位置 (相手陣地の最奥) に到達したとき、そのポーンをキング以外の別の駒に変化させることができる。これをプロモーション (昇格)(Promotion) と言う。

通常のチェスに存在するアンパッサン (En passamnt)(ポーンの特殊な動き)、キャスリング (Casling)(キングとルークの特殊な動き) は、ラオムシャッハにはない。

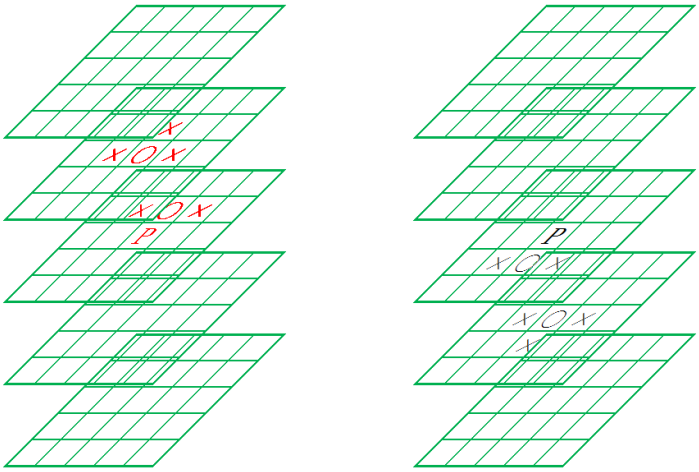

図 4　ポーンの動き方 (左は白、右は黒)

他、ラオムシャッハでは兵力不足 (どちらのプレイヤーもキングのみになった場合等、決着をつけることが不可能になった)、千日手 (同じ局面を繰り返す)、50 手ルール (50 手に渡り一度もポーンが動かず駒取りもない) 等も、通常のチェスと同様引き分けとなるが、本研究で開発したアプリケーションにはそれらの機能は搭載されていない。

3　アプリケーション

本章では、本研究で作成したラオムシャッハアプリケーションについて述べる。Windows 上で実行できるアプリケーションには、Windows アプリケーションとコンソールアプリケーションの 2 種類がある。Windows アプリケーションとは GUI(グラフィカルユーザインタフェース) で動作するウィンドウアプリケーションのことで、操作、実行結果などをグラフィカルに表現することができるアプリケーションである。一方、コンソールアプリケーションとは CUI(コマンドラインインタフェース) で動作するアプリケーションで、実行結果は文字列での出力となる。本研究で作成するアプリケーションは、ゲームアプリケーションであることからグラフィカルな表現の可能な Windows アプリケーションを用いる。

3.1　Windows アプリケーションの基礎

Windows アプリケーションはコンソールアプリケーションとは異なりイベント駆動という作法に従ってプログラムされる。イベント駆動とは、OS から送られてくるイベントメッセージを受け取り、それに応じて処理を実行する形式である。イベントメッセージの例としては WM_PAINT(表示処理)、WM_DESTROY(終了処理) 等がある。また Windows アプリケーションの作成には Win32 API の関数を用いるため、ヘッダファイル windows.h をインクルードする必要がある。

表 2 に代表的な Windows アプリケーションのヘッダファイルを示す。また、表 3 に代表的な Windows アプリケーションの関数を示す。以下に代表的なヘッダおよび代表的な関数について述べる。

- tchar.h

表 2　代表的なヘッダファイル

ヘッダファイル	用途
windows.h	WIN32 API 関数の使用
tchar.h	文字コードへの対応

表 3　代表的な関数

関数	用途
WinMain	Win アプリケーションプログラムの main 関数
CreateWindow	ウィンドウの作成
ShowWindow	作成されたウィンドウの表示
PeekMessage	Windows(OS) からのイベントメッセージの受け取り
TranslateMessage	PeekMessage 関数が受け取ったメッセージを適当な形に変換
DispatchMessage	ウィンドウプロシージャに TranslateMessage 関数が作成したメッセージを送出
ウィンドウプロシージャ	DispatchMessage 関数からのメッセージを受け取り処理を実行
RegisterClassEx	ウィンドウクラスの登録
UnregisterClass	ウィンドウクラスの登録解除
ZeroMemory	指定されたアドレスを 0 で初期化する
ValidateRect	描画の終了を Windows に伝える

tchar.h を用いることで文字コードに関係無く関数を用いることができるようになる。旧型の Windows や MS-DOS では異なるバイト数の文字コードが混在して使用されていた (MBCS:MultiByte Character Set) ため、Win32 API の関数等には近年のユニコード用の関数と MBCS 用の関数の二種類が用意されているものが多い。それらを意識せずに使用するためにヘッダファイル tchar.h をインクルードして TCHAR マクロを使用する。このマクロによってユニコード、MBCS 両方に対応したソースコードを作成することができる。使用方法は文字列リテラルを_T() で囲むことである。

- WinMain 関数

Windows アプリケーションでは main 関数ではなく WinMain 関数からプログラムが開始される。以下に WinMain 関数のプロトタイプ宣言を示す。

```
int WINAPI WinMain(
   HINSTANCE hIstance,
   HINSTANCE hPrevInstance,
   LPSTR lpCmdLine,
   int nCmdShow
);
```

最初の int の後の WINAPI は呼び出し規約であるが#define プリプロセッサにより_stdcall に置き換えられ

る。第一、第二引数の HINSTANCE はインスタンスハンドルのための型である。これは Windows 上で並列に起動させているプログラムごとの区別に用いられる。なお、第二引数の hPrevInstance は旧型の Windows との互換性のために残されているもので、その値は常に NULL である。第三引数の lpCmdLine はコマンドライン引数のアドレスを格納する引数、最後の引数 nCmdShow はウィンドウの状態を表す定数が入る。

- CreateWindow 関数
- ShowWindow 関数

CreateWindow 関数と ShowWindow 関数はウィンドウを表示するのに用いられる関数である。以下に CreateWindow 関数と ShowWindow 関数のプロトタイプ宣言を示す。

```
HWND CreateWindow(
   LPCTSTR lpClassName,
   LPCTSTR lpWindowName,
   DWORD dwStyle,
   int x,y,
   int nWidth,nHeight,
   HWND hWndParent,
   HMENU hMenu,
   HINSTANCE hInstance,
   LPVOID lpParam
);

BOOL ShowWindow(
   HWND hWnd,
   int nCmdShow
);
```

まず CreateWindow 関数の引数を説明する。lpClassName は登録されているクラス名,lpWindowName はウィンドウ名,dwStyle はウィンドウスタイル, x と y はウィンドウの位置,nWidth と nHeight はウィンドウのサイズ,hWndParent は親ウィンドウのハンドル, hMenu はメニューハンドル,hInstance はアプリケーションインスタンスのハンドル,lpParam はウィンドウ作成データをそれぞれ取る。

次に ShowWindow 関数の引数を説明する。hWnd はウィンドウのハンドル、nCmdShow は表示状態をそれぞれ取る。

- PeekMessage 関数
- TranslateMessage 関数
- DispatchMessage 関数

ウィンドウを表示させ続けるためにはプログラム内にメッセージループを作成する必要がある。メッセージループは Windows から送られてくるイベントメッセージを監視するためのループであり、PeekMessage 関数 (Windows からのイベントメッセージの受け取り)、TranslateMessage 関数 (WM_KEYDOWN メッセージから WM_CHAR メッセージを作成)、DispatchMessage 関数 (ウィンドウプロシージャにイベントメッセー

ジを転送) を組み合わせて書く。以下に PeekMessage 関数, TranslateMessage 関数, DispatchMessage 関数のプロトタイプ宣言を示す。

```
BOOL PeekMessage(
   LPMSG lpMsg,
   HWND hWnd,
   UINT wMsgFilterMin,
   UINT wMsgFilterMax,
   UINT wRemoveMsg
);
```

lpMsg はメッセージ情報のポインタ、hWnd はウィンドウのハンドル、wMsgFilterMin と wMsgFilterMax は最初と最後のメッセージ (通常は 0L)、wRemoveMsg は削除オプションである。

```
typedef struct tagMSG(
   HWND hWnd;
   UINT message;
   WPARAM wParam;
   LPARAM lParam;
   DWORD time;
   POINT pt;
)MSG;
```

hWnd はウィンドウのハンドラ、message はメッセージ、wParam と lParam はパラメータ、time はメッセージの送信時間、pt は送信時のマウスカーソル位置である。

```
BOOL TrandlateMessage(
   CONST MSG *lpMsg
);

BOOL DispatchMessage(
   CONST MSG *lpmsg
);
```

lpMsg と lpmsg はどちらもメッセージ情報である。

- ウィンドウプロシージャ

ウィンドウプロシージャとはウィンドウと結び付けられた特殊な関数で、イベントメッセージに合わせた処理を書いておくことで DispatchMessage 関数からのメッセージを処理することができる。

```
LRESULT WINAPI WindowProc(
   HWND hWnd,
   UINT msg,
   WPARAM wParam,
```

```
   LPARAM lParam
);
```

以上がウィンドウプロシージャのプロトタイプ宣言である。ウィンドウプロシージャの関数名は自由だが、引数と返値の型は宣言通りでなくてはいけない。hWnd はウィンドウのハンドル、msg はメッセージの識別子、wParam と lParam はメッセージパラメータである。

そして最後にウィンドウとウィンドウプロシージャを関連付けるウィンドウクラスを登録する。

- RegisterClassEx 関数
- UnregisterClass 関数

RegisterClassEx 関数はウィンドウクラスを登録するための関数である。ウィンドウクラスを登録するには、WNDCLASSEX 構造体に必要な情報を入れておき RegisterClassEx 関数で Windows に登録する。プログラムを終了するには UnregisterClass 関数で登録を解除する必要がある。以下に RegisterClassEx 関数, UnregisterClass 関数のプロトタイプ宣言を示す。

```
ATOM RegisterClassEx(
   CONST WNDCLASSEX *lpwcx
);

typedef struct tagWNDCLASSEXW(
   UINT cbSize;
   UINT style;
   WNDPROC lpfnWndProc;
   int cbClsExtra;
   int cbWndExtra;
   HINSTANCE hInstance;
   HICON hIcon;
   HCURSOR hCursor;
   HBRUSH hbrBackground;
   LPCWSTR lpszMenuName;
   LPCWSTR lpszClassName;
   HICON hIconSm;
);WNDCLASSEX;

BOOL UnregisterClass(
   LPCTSTR lpClassName,
   HINSTANCE hInstance
);
```

大量のメンバ・引数があるが、重要なもののみを説明する。lpClassName はウィンドウクラス名、lpfnWndProc はウィンドウプロシージャのアドレス、style はウィンドウクラスのスタイルを取る。

- ZeroMemory 関数

表 4　代表的なコンポーネント

コンポーネント	用途
DirectX Graphics	画面描画に関するコンポーネント
DirectX Audio	音楽再生に関するコンポーネント
DirectX Input	入力機器に関するコンポーネント
DirectX Play	ゲーム用のネットワーク通信に関するコンポーネント
DirectX Show	動画・音声再生に関するコンポーネント (Windows SDK に移行済み)

ZeroMemory 関数は渡された変数を 0 で初期化する関数である。第一引数に変数のアドレス、第二引数に変数のバイト数を指定する。以下に ZeroMemory 関数のプロトタイプ宣言を示す。

```
void ZeroMemory(
   PVOID Destination,
   SIZE_T Length
);
```

- ValidateRect 関数

ValidateRect 関数は描画が終了したことを OS に伝えるための関数である。これを呼び出すとしばらく WM_PAINT メッセージが送られてこなくなる。この関数は後ほど DirectX による描画を行う際に使用する。以下に ValidateRect 関数のプロトタイプ宣言を示す。

```
BOOL ValidateRect(
   HWND hWnd,
   const RECT *lpRect
);
```

ここまでの関数をまとめて使用したスケルトンプログラムを付録 A に示す。

3.2　DirectX で使用した機能

DirectX[3] は Windows の機能の一部で、マルチメディア処理を担当するソフトウェアである。DirectX は Windows 95 から搭載されたソフトウェアであったが、Windows Vista では通常のウィンドウ表示にも使われる必須コンポーネントとなっている。DirectX は担当分野ごとに幾つかのコンポーネントに分かれている。

表 4 に代表的な DirectX のコンポーネントを示す。本研究では画面描画用のコンポーネントである DirectX Graphics、入力機能に関するコンポーネントである DirectX Input を利用してアプリケーションを作成した。

DirectX の関数を利用するには通常のプログラムと同様ヘッダファイルのインクルードが必要である。DirectX Graphicshs の d3dx9.h、DirectX Input の dinput.h を今回はインクルードすることになる。しかしここで注意することがある。C 言語の標準ライブラリ等のコードは、ソースコードをビルドした時点で実行ファイル (EXE ファイル) に取り込まれる。この方式をスタティックリンク (静的リンク) という。一方、DirectX のライブラリは実行ファイルの外部に別ファイル (dll(ダイナミックリンクライブラリ) ファイル) と

表 5　COM インタフェース

COM インタフェース	用途
IDirect3D9	GPU の性能を確認したり他のインタフェースのインスタンスの作成をする
IDirect3DDevice9	実際の描画処理を行うメソッドを持つ

表 6　IDirect3D9 の関数

関数	用途
CreateDevice	IDirect3DDevice9 のインスタンスを作成する

して、実行中にそれらのファイルと通信しあって処理を行う。これをダイナミックリンク (動的リンク) という。このことからヘッダファイルのインクルードだけでなく、ライブラリファイルの依存関係の登録も必要である。ダイナミックリンク形式のプログラムの利点は実行ファイルが小さくなることや、ライブラリだけのバージョンアップが可能なことなどが挙げられる。EXE ファイルと DLL ファイルが通信する仕組みを COM(Component Object Model) という。ライブラリの関数はそのほとんどが COM インタフェースと呼ばれる構造体のようなもののメンバーにまとめられているため、DirectX の関数を利用するには最初に COM インタフェースのインスタンスを作成する必要があり、インスタンスは終了時に Release 関数によって消去する必要がある。

表 5 に使用した COM インタフェースを示す。

- IDirect3D9

IDirect3D9 は GPU の性能を調べたり他の COM インタフェースのインスタンスの作成などの DirectX を扱う上での準備作業を行う COM インタフェースである。そのため他のインタフェースよりも先にインスタンスを作成する必要がある。IDirect3D9 は Direct3DCreate9 関数を使って作成される。以下に Direct3DCreate9 関数のプロトタイプ宣言を示す。

```
IDirect3D9 *Direct3DCreate9(
   UINT SDKVersion
);
```

なお、Direct3DCreate9 関数の引数はバージョンチェックのためのものなので常に D3D_SDK_VERSION を指定する。

表 6 に本研究で使用した IDirect3D9 の持つ関数を示す。

- CreateDevice 関数

CreateDevice 関数は IDirect3DDevice9 のインスタンスを作成する関数である。以下に CreateDevice 関数の宣言を示す。

```
HRESULT CreateDevice(
   UINT Adapter,
```

表 7　IDirect3DDevice9 の関数

関数	用途
Clear	バックバッファの消去
BeginScene	描画開始の通知
EndScene	描画終了の通知
Present	バックバッファをフロントバッファにスワップ

```
    D3DDEVTYPE DeviceType,
    HWND hFocusWindow,
    DWORD BehaviorFlags,
    D3DPRESENT_PARAMETERS *pPresentationParameters,
    IDirect3DDevice9** ppReturnedDeviceInterface
);
```

Adapter は使用するディスプレイアダプタ、DeviceType はデバイスタイプ、hFocusWindow は描画対象となるウィンドウのハンドル、BehaviorFlags はデバイスのオプション、*pPresentationParameters はパラメータ構造体のポインタ、ppReturnedDeviceInterface はデバイスを返すポインタを表す。

- IDirect3DDevice9

IDirect3DDevice9 は実際の描画作業を行う関数の大半を持つ COM インタフェースである。IDirect3D9 の CreateDevice 関数によって作成される。作成される際に D3DPRESENT_PARAMETERS 構造体を利用して描画属性を設定される。D3DPRESENT_PARAMETERS 構造体は描画対象の属性をまとめておく構造体である。以下に D3DPRESENT_PARAMETERS 構造体のプロトタイプ宣言を示す。

```
typedef struct _D3DPRESENT_PARAMETERS_
{
    UINT                BackBufferWidth;
    UINT                BackBufferHeight;
    D3DFORMAT           BackBufferFormat;
    UINT                BackBufferCount;
    D3DMULTISAMPLE_TYPE MultiSampleType;
    DWORD               MultiSampleQuality;
    D3DSWAPEFFECT       SwapEffect;
    HWND                hDeviceWindow;
    BOOL                Windowed;
    BOOL                EnableAutoDepthStencil;
    D3DFORMAT           AutoDepthStencilFormat;
    DWORD               Flags;
    UINT                FullScreen_RefreshRateInHz;
```

```
    UINT                PresentationInterval;
} D3DPRESENT_PARAMETERS;
```

BackBufferWidth、BackBufferHeight、BackBufferFormat、BackBufferCount はそれぞれバックバッファの幅と高さとフォーマットと数、MultiSampleType、MultiSampleQuality はマルチサンプリングの方式と品質、SwapEffect は切り替え後のバッファの処理方法、hDeviceWindow は評するウィンドウのハンドル、Windowed はフルスクリーンか否か、EnableAutoDepthStencil、AutoDepthStencilFormat 深度ステンシルバッファの有効無効と形式、Flags はバックバッファに関するフラグ、FullScreen_RefreshRateInHz はフルスクリーンの時のリフレッシュレート、PresentationInterval はバッファ切り替えタイミングの設定である。

- Clear 関数
- BeginScene 関数
- EndScene 関数
- Present 関数

DirectX は描画を行うためにバックバッファ (裏画面) を何枚か用意して、その内の一枚を実際に出力されるフロントバッファにする。描画処理はバックバッファで行われ、順にフロントバッファをスワップすることで描画中のチラツキを見えなくしている。

バックバッファへの描画の順序は Clear 関数でバックバッファの内容を消去し、BeginScene 関数で描画の開始を通知して、一画面分の描画を行う。そして EndScene 関数で描画終了を通知し、Present 関数でフロントバッファとバックバッファをスワップする。これが DirectX による描画の基本となる。

以下に各関数の宣言を示す。

```
HRESULT Clear(          DWORD Count,
    const D3DRECT *pRects,
    DWORD Flags,
    D3DCOLOR Color,
    float Z,
    DWORD Stencil
);

HRESULT BeginScene(VOID);

HRESULT EndScene(VOID);

HRESULT Present(          CONST RECT *pSourceRect,
    CONST RECT *pDestRect,
    HWND hDestWindowOverride,
    CONST RGNDATA *pDirtyRegion
);
```

ここまでの説明は DirectX の描画機能に関する説明である。ここからは DirectX による描画の内容について説明する。

DirectX が 3D モデルを描画するには 3D モデルを用意し、それをワールド空間 (コンピュータ内の仮想三

表 8　3D モデルの読み込みと表示

関数	用途
D3DXLoadMeshFromX	X ファイルからメッシュデータを読み込む
D3DXCreateTextureFromFile	画像ファイルからテクスチャデータを読み込む
SetMaterial	デバイスにマテリアルを設定
SetTexture	デバイスにテクスチャを設定
DrawSubset	メッシュを描画

次元空間) に配置し、射影変換により 2D 化した後、ラスタライズ (ピクセルデータ化) を行う必要がある。3D モデルは 3〜4 個の頂点をつないでポリゴン (面) を作り、それを組み合わせて作成する。つまり 3D モデルは頂点データの集まりである。DirectX で 3D モデルを扱うには二種類の方法がある。一つは頂点バッファを用いる方法。これは作成時にどんな情報が必要かを指定する必要があるのだが、頂点データと一言に言っても座標・法線・頂点色等様々な種類の情報があるため、自由な形状のモデルを作りにくく、円や立方体といった幾何学的なモデルの作成に向いている。もう一つはメッシュ (Mesh) という 3D モデリングソフトなどで作成したモデルファイルを読み込む方法である。本研究ではメッシュを利用してアプリケーションを作成したためメッシュの説明を行う。

表 8 に本研究で使用した 3D モデルの読み込みと表示に関する関数を示す。

- D3DXLoadMeshFromX 関数
- D3DXCreateTextureFromFile 関数

D3DXLoadMeshFromX 関数は X ファイルからメッシュデータを読み込み、ID3DXMesh インタフェースのインスタンスに格納する。マテリアル (材質データ) は D3DMATERIAL9 構造体に格納する。D3DXCreateTextureFromFile 関数は画像ファイルからテクスチャデータを作成し、IDirect3DTexture9 インタフェースのインスタンスに格納する。マテリアルとテクスチャはひとつのモデルに対し複数含まれることがあるため、配列として扱えるようになっている。以下に各関数の宣言を示す。

```
HRESULT D3DXLoadMeshFromX(          LPCTSTR pFilename,
    DWORD Options,
    LPDIRECT3DDEVICE9 pDevice,
    LPD3DXBUFFER* ppAdjacency,
    LPD3DXBUFFER* ppMaterials,
    LPD3DXBUFFER* ppEffectInstances,
    DWORD* pNumMaterials,
    LPD3DXMESH* ppMesh
);

HRESULT D3DXCreateTextureFromFile(          LPDIRECT3DDEVICE9 pDevice,
    LPCTSTR pSrcFile,
    LPDIRECT3DTEXTURE9 *ppTexture
```

表 9　3D モデルの読み込みと表示

関数	用途
SetTransform	デバイスに座標変換行列を設定
SetLight	デバイスにライトを設定
D3DXMatrixLookAtLH	デバイスにカメラを設定
D3DXMatrixPerspectiveFovLH	射影変換行列の設定

```
);
```

pFilename は読み込む X ファイルの名前、Options はメッシュのメモリへの格納方法、pDevice は Direct3D デバイスのポインタ、ppAdjacency は隣接性データの記録ポインタ、ppMaterials はマテリアルデータへのポインタ、ppEffectInstances はエフェクトインスタンスのポインタ、pNumMaterials はマテリアルの数を記録するポインタ、ppMesh はメッシュデータを記録するポインタである。

pDevice は DirectX3D デバイス、pSrcFile はテクスチャファイル名、ppTexture は読み込んだテクスチャを返すポインタである。

- SetMaterial 関数
- SetTexture 関数
- DrawSubset 関数

SetMaterial 関数は IDirect3DDevice9 の関数でデバイスにマテリアルを設定する。SetTexture 関数は IDirect3DDevice9 の関数でデバイスにテクスチャを設定する。DrawSubset 関数は ID3DXMesh の関数でメッシュの描画を行う。以下に各関数の宣言を示す。

```
HRESULT SetMaterial(          CONST D3DMATERIAL9 *pMaterial
);

HRESULT SetTexture(          DWORD Stage,
    IDirect3DBaseTexture9 *pTexture
);

HRESULT DrawSubset(          DWORD AttribId
);
```

次にワールド空間内のライト (光源) とカメラ (視点) の設定を説明する。表 9 に本研究で使用したライトとカメラに関する関数を示す。

- SetTransform 関数

SetTransform 関数はデバイスに行列を渡す関数である。この関数はワールド変換行列、ビュー変換行列、射影変換行列の設定に使用される。以下に SetTransform 関数の宣言を示す。

```
HRESULT SetTransform(          D3DTRANSFORMSTATETYPE State,
```

```
    CONST D3DMATRIX *pMatrix
);
```

state は行列の種類を指定する定数、pMatrix は設定する行列である。state に D3DTS_WORLD を指定するとワールド変換行列、D3DTS_VIEW ならばビュー変換行列、D3DTS_PROJECTION ならば射影行列が設定される。また行列を格納する D3DMATRIX 構造体の宣言を以下に示す。

```
typedef struct _D3DMATRIX {
    union {
        struct {
            float        _11, _12, _13, _14;
            float        _21, _22, _23, _24;
            float        _31, _32, _33, _34;
            float        _41, _42, _43, _44;

        };
        float m[4][4];
    };
} D3DMATRIX;
```

ワールド空間における拡大縮小・移動・回転は全て４行４列の行列を使用する。

- SetLight 関数

SetLight 関数はデバイスに光源の向き、種類などを D3DLIGHT9 構造体に設定し LightEnable 関数で有効にすることで、証明を設定する。以下に SetLight 関数と D3DLIGHT9 構造体の宣言を示す。

```
HRESULT SetLight(          DWORD Index,
    CONST D3DLIGHT9 *pLight
);
```

Index はライトのインデックス、pLight は D3DLIGHT9 構造体へのポインタである。

```
typedef struct _D3DLIGHT9 {
    D3DLIGHTTYPE Type;
    D3DCOLORVALUE Diffuse;
    D3DCOLORVALUE Specular;
    D3DCOLORVALUE Ambient;
    D3DVECTOR Position;
    D3DVECTOR Direction;
    float Range;
    float Falloff;
    float Attenuation0;
    float Attenuation1;
```

```
    float Attenuation2;
    float Theta;
    float Phi;
} D3DLIGHT9;
```

Type は光源の種類、Diffuse はディフューズ色、Specular はスペキュラ色、Ambient はアンビエント色、Position は位置、Direction は向き、Range は有効距離、Falloff はスポットライトの内部と外部の輝度調整 Attenuation0 と Attenuation1 と Attenuation2 は減衰定数、Theta はスポットライトの内部コーンの角度、Phi はスポットライトの外部コーンの角度である。

```
HRESULT LightEnable(          DWORD LightIndex,
    BOOL bEnable
);
```

LightIndex に操作対象のライトのインデックス、bEnable を TRUE にすることでライトが点灯する。

- D3DXMatrixLookAtLH 関数
- D3DXMatrixPerspectiveFovLH 関数

D3DXMatrixLookAtLH 関数と D3DXMatrixPerspectiveFovLH 関数はどちらもカメラに関する関数である。以下に宣言を示す。

```
D3DXMATRIX *D3DXMatrixLookAtLH(          D3DXMATRIX *pOut,
    CONST D3DXVECTOR3 *pEye,
    CONST D3DXVECTOR3 *pAt,
    CONST D3DXVECTOR3 *pUp
);
```

ｐ Out は結果を返すポインタ、pEye はカメラの位置座標、pAt は注視点の座標、pUp は上方向を表すベクトルである。つまりワールド空間で座標 pEye から座標 pAt を見た視点が画面に描画されることになる。

```
D3DXMATRIX *D3DXMatrixPerspectiveFovLH(          D3DXMATRIX *pOut,
    FLOAT fovY,
    FLOAT Aspect,
    FLOAT zn,
    FLOAT zf
);
```

ｐ Out は結果を返すポインタ、fovy は y 方向の視野角、Aspect は画面のアスペクト比 (幅÷横)、zn と zf はそれぞれ近遠クリップ面の Z 値を表すベクトルである。

最後に文字の描画について説明する。DirrectX で文字を表示するには ID3DXFont インタフェースを使用する。ID3DXFont は 2D グラフィックス描画の仕組みであるスプライトを利用して、ウィンドウの最前面に文字を貼り付けることができる。表 10 に本研究で使用した文字の描画に関するインスタンス・関数を示す。なお、これらのインスタンスも終了時に消去する必要がある。

表 10　3D モデルの読み込みと表示

インスタンス	用途
ID3DXFont	フォントのレンダリングを行うインタフェース
ID3DXSprite	スプライトのインスタンス
関数	用途
Begin	スプライトを開始する
DrawText	テキストをレンダリングしてスプライトに貼り付ける
End	スプライトを終了する

- ID3DXFont
- ID3DXSprite

ID3DXFont はフォントファイルからデータを読み取りフォントデータをビデオメモリ内に用意する。このためインスタンスごとにフォントの種類・大きさが決められることになり、複数のフォントを混在させるためにはインスタンスも複数作成する必要がある。インスタンスの作成は D3DXCreateFont 関数によって行われる。ID3DXSprite はスプライトのインスタンスである。スプライトは 3D のワールド空間が表示された画面の上にセロハンテープを貼りつけるようなものだと考えるとわかりやすい。これに ID3DXFont が用意したフォントデータを書きこむことで、文字の描画を行うことができる。インスタンスの作成は D3DXCreateSprite 関数によって行われる。以下に D3DXCreateFont 関数と D3DXCreateSprite 関数の宣言を示す。

```
HRESULT D3DXCreateFont(
  LPDIRECT3DDEVICE9 pDevice,
  INT Height,
  UINT Width,
  UINT Weight,
  UINT MipLevels,
  BOOL Italic,
  DWORD CharSet,
  DWORD OutputPrecision,
  DWORD Quality,
  DWORD PitchAndFamily,
  LPCTSTR pFacename,
  LPD3DXFONT * ppFont
);

HRESULT D3DXCreateSprite(          LPDIRECT3DDEVICE9 pDevice,
    LPD3DXSPRITE *ppSprite
);
```

pDevice はどちらも IDirect3DDevice9 へのポインタ、Height と Width と Weight はそれぞれフォント

の高さ・幅・太さ、MipLevels はミップマップレベル、Italic は斜体の有効無効、CharSet は文字セット、OutputPrecision と Quality はフォントの選択方法と品質 PitchAndFamily はフォントのピッチとファミリ、pFacename はフォント名、ppFont はフォントのインスタンスのポインタ、ppSprite は ID3DXSprite のポインタである。

- Begin 関数
- End 関数

Begin 関数と End 関数は ID3DXSprite の関数で、スプライトの描画開始と描画終了を行う関数である。以下に各関数の宣言を示す。

```
HRESULT Begin(
  DWORD Flags
);
```

Flags はスプライトのレンダリングオプションである。

```
HRESULT End();
```

- DrawText 関数

DrawText 関数は ID3DXFont の関数で、スプライトへのテキスト描画を行う。以下に宣言を示す。

```
INT DrawText(
  LPD3DXSPRITE pSprite,
  LPCTSTR pString,
  INT Count,
  LPRECT pRect,
  DWORD Format,
  D3DCOLOR Color
);
```

pSprite はスプライトのポインタ、pString は描画する文字列、Count は文字数、pRect が描画領域、Format は文字の成形方法、Color は描画色である。

以上が画面描画とその内容に関する説明である。最後に入力の検出に関して説明する。DirectX では Direct Input コンポーネントを利用することでマウス、キーボード、ジョイパッド等の入力を検知する事ができる。本研究ではキーボード入力を検知しアプリケーションを操作する機能を実装した。表 11 に本研究で使用した Direct Input の持つ関数を示す。

- DirectInput8Create 関数

IDirectInput8 と IDirectInputDevice8 は DirectX Graphics の IDirect3D9 と IDirect3DDevice9 とほぼ同じ関係にある。以下に各関数の宣言を示す。

```
HRESULT WINAPI DirectInput8Create(          HINSTANCE hinst,
    DWORD dwVersion,
```

表 11　入力に関する関数

関数	用途
DirectInput8Create	IDirectInput8 インタフェースのインスタンスを作成する
CreateDevice	IDirectInputDevice8 インタフェースのインスタンスを作成する
GetDeviceState	キーボードの入力状態をチェックする

```
    REFIID riidltf,
    LPVOID *ppvOut,
    LPUNKNOWN punkOuter
);
```

hinst はアプリケーションのインスタンスハンドル、dwVersion はバージョン、riidltf は目的のインターフェイスの識別子、ppvOut はポインタのアドレス、punkOuter はインターフェイスのアドレスポインタである。

```
HRESULT CreateDevice(           REFGUID rguid,
    LPDIRECTINPUTDEVICE *lplpDirectInputDevice,
    LPUNKNOWN pUnkOuter
);
```

rguid がデバイスの識別子となる。(GUID_SysKeyboard がデフォルトのキーボード) また、IDirectInputDevice8 は SetDataFormat 関数と SetCooperativeLevel 関数を使ってデータの形式と強調レベルを設定する必要がある。

```
HRESULT GetDeviceStatus(          REFGUID rguidInstance
);
```

GetDeviceStatus 関数は 256 バイトの配列変数で現在のキーボードの状態を取得する関数である。メッセージループの最初に実行することで処理の分岐をキーボード入力に対応させることができる。

3.3　ラオムシャッハプログラム

本節では、本研究で作成したラオムシャッハプログラムについて説明する。付録 B にラオムシャッハプログラムを示す。なお、本アプリケーションは main.cpp、game.cpp、my3dlib.cpp、my3dlib.h の 4 つのファイルと各種モデルデータで形成されている。my3dlib.cpp と my3dlib.h はどちらもここまでで説明した C 標準ライブラリや DirectX の関数を簡単に扱うために用意したライブラリである。main.cpp にはメッセージループやゲームの実際の処理等、game.cpp には駒の移動範囲の判定等の細かい処理をまとめている。

Windows アプリケーションはプログラム内でメッセージループを作り、描画を繰り返していることをここまでで説明した。メッセージループによる描画を常に繰り返している状態では、分岐、もしくはループで何らかの入力を待機するといった処理を作るのは難しい。そのため本アプリケーションではチェスのゲームの状態を表 12 に示す 4 つの状態に分けて考え、状態をグローバル変数を参照することで確認できるようにしている。

表 12　ゲームの状態

	ゲームの状態
1	動かす駒を選択する
2	移動・攻撃するマスを選択する
3	プロモーションを行う
4	ゲームを終了し勝敗を表示する

表 13　ゲームの状態

関数	機能
ShowCursor	マウスカーソルの表示と非表示
InitD3DWindow	ウィンドウの準備と DirectX の準備
LoadModels	モデルデータの読み込み
SetViews	カメラ・ライトの初期設定
Render	画面描画
GameMain	ゲーム本体
CleanupD3D	各インスタンスの解放
reset	チェス盤の初期化
moveCheck	移動範囲の検索
checkCheck	チェックが起こるか否か
checkmateCheck	チェックメイトか否か
stalemateCheck	ステールメイトか否か
promotionCheck	プロモーションか否か

次に表 13 に本アプリケーションで使用されている関数の機能を示す。
以下に各関数の説明を示す。

- ShowCursor 関数

ShowCursor 関数はウィンドウ上でのマウスカーソルの表示非表示を切り替える関数である。この関数は Windows.h からインクルードされている。以下に関数の宣言を示す。

```
int ShowCursor(
  BOOL bShow   // カーソルの可視状態
);
```

hShow が TRUE ならばマウスカーソルを表示、FALSE なら非表示である。

- InitD3DWindow 関数

InitD3DWindow 関数はウィンドウの作成、表示、ウィンドウプロシージャの作成、ウィンドウクラスの登録、DirectX の各種インスタンスの作成等の描画準備をすべて行う関数である。my3dlib.cpp にライブラリ化

された関数であり、アプリケーションの準備作業を簡単に行うことができるようにしている。

- LoadModels 関数

LoadModels 関数はモデルの読み込みを行う各種関数をまとめた関数である。

- SetViews 関数

SetViews 関数はライトの設定とカメラの初期位置の設定を行う関数である。

- Render 関数

Render 関数は描画を行う関数である。メッセージループで常に呼び出されるため、この中にゲームの本体の関数を置くことで、ゲームプログラムは動作している。

- GameMain 関数

GameMain 関数はゲームの本体に当たる関数である。この関数が以降の関数を呼び出すことでゲームは進行する。

- CleanupD3D 関数

CleanupD3D 関数は各種インスタンスの解放を行う関数である。この関数はウィンドウプロシージャが WM_DESTROY メッセージを受け取った時、つまりウィンドウが閉じられたときに呼び出される。

- reset 関数

reset 関数はラオムシャッハのチェス盤の内容を表すグローバル変数として用意された int 型の三次元配列変数 board の内容をリセットし、ラオムシャッハの初期配置を行う関数である。

- moveCheck 関数

moveCheck 関数は駒を選択する状態の時に呼び出され、選択されている駒の移動・攻撃を行うことのできるマスを、グローバル変数として用意された int 型の三次元配列変数 tmp に格納する関数である。

- checkCheck 関数

checkCheck 関数はグローバル変数 board を参照しキングに対してチェックが起こっているときに true を返す関数である。

- checkmateCheck 関数

checkmateCheck 関数はグローバル変数 board を参照しキングに対してチェックメイトが起こっているときに true を返す関数である。

- stalemateCheck 関数

stalemateCheck 関数はグローバル変数 board を参照しステールメイトが起こっているときに true を返す関数である。

- promotionCheck 関数

promotionCheck 関数はグローバル変数 board を参照しプロモーションが起こるときに true を返す関数である。

3.4 プログラムの仕様

本節では本研究で作成したアプリケーションの簡単な仕様を説明する。アプリケーションは実行ファイル 3D-Chess.exe を実行する (エクスプローラ上でダブルクリック等) ことで起動する。本アプリケーションは起動した時点でマウスカーソルを非表示にする。そしてウィンドウの準備、DirectX のインターフェースの準備、モデルのロード、チェス盤の初期配置を行った後、メッセージループに入る。起動画面を図 5 に示す。

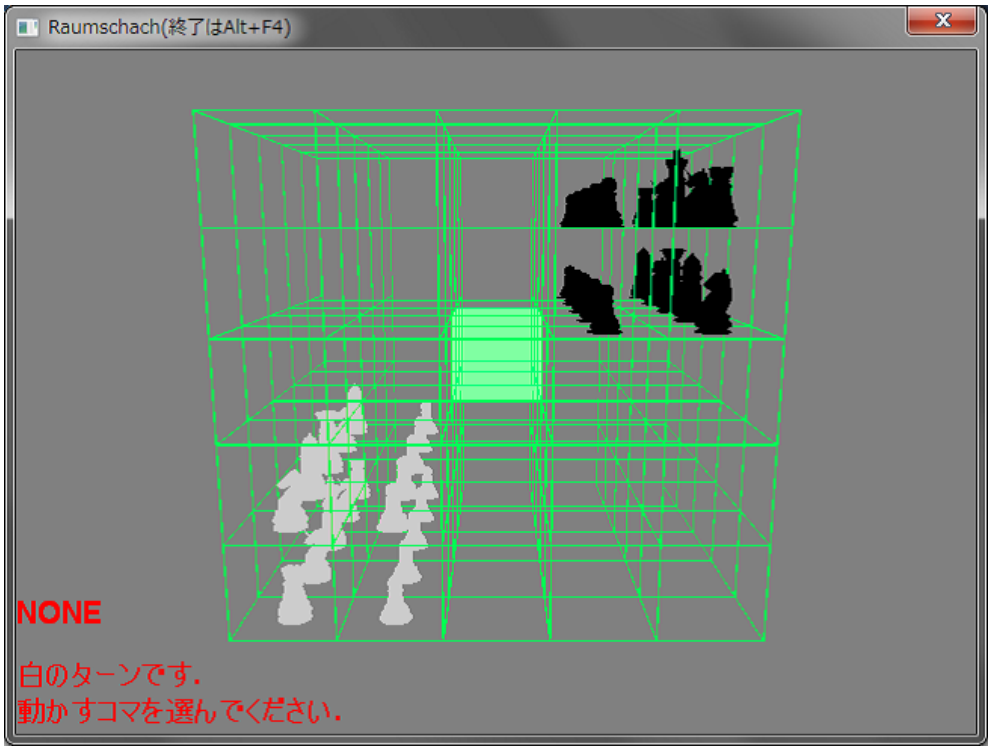

図 5　アプリケーションの初期画面

本アプリケーションは、起動時には動かす駒の入力待ち状態となっている。先手は白であるので、まず白駒プレイヤーが第一手で動かす駒を選択する。手番となったプレイヤーは、以下の 3 つの操作を行う。

1. 移動する駒を選択
   緑色のボックスとして表示されるカーソルを動かして移動する駒を選択する。プレイヤーはキーボードの上下左右キーと/キー・\ キーを入力することでカーソルを移動することができる。カーソルが動かしたい駒を指しているときに Enter キーを入力すると、選択した駒が移動できる範囲が青色のボックスで、攻撃できるマスが赤色のボックスで表示される。
2. 選択した駒を移動する先を選択

選択した駒を移動させたい位置に上下左右キーと/キー・\ キーでカーソルを動かし、Enter キーを入力すると、カーソルが指すマスに駒が移動する。このとき、移動・攻撃できないマスを選択すると自分の手番のまま 1. に戻る。

3. ポーンが昇格する駒の選択
   ポーンが最前列に到達した場合、昇格したい駒の記号 (Q・N・B・R・U) をキーボードで入力するとポーンがその駒に変化する。

手番となったプレイヤーが上記の操作を終えると、相手に手番が移る。また、チェックメイト、ステールメイトとなった場合、勝敗が表示されゲーム終了となる。ステップ 1. で駒を選択した画面を図 6 に示す。図 6 に示されるように、選択した駒が移動できるマス、攻撃できるマスがそれぞれ青、赤で表示される。また、手番中に Shift キーを入力しながら、上下左右キーと/キー・\ キーを入力することでカメラを移動することができる。表 14 に操作に使用するキーの一覧を示す。

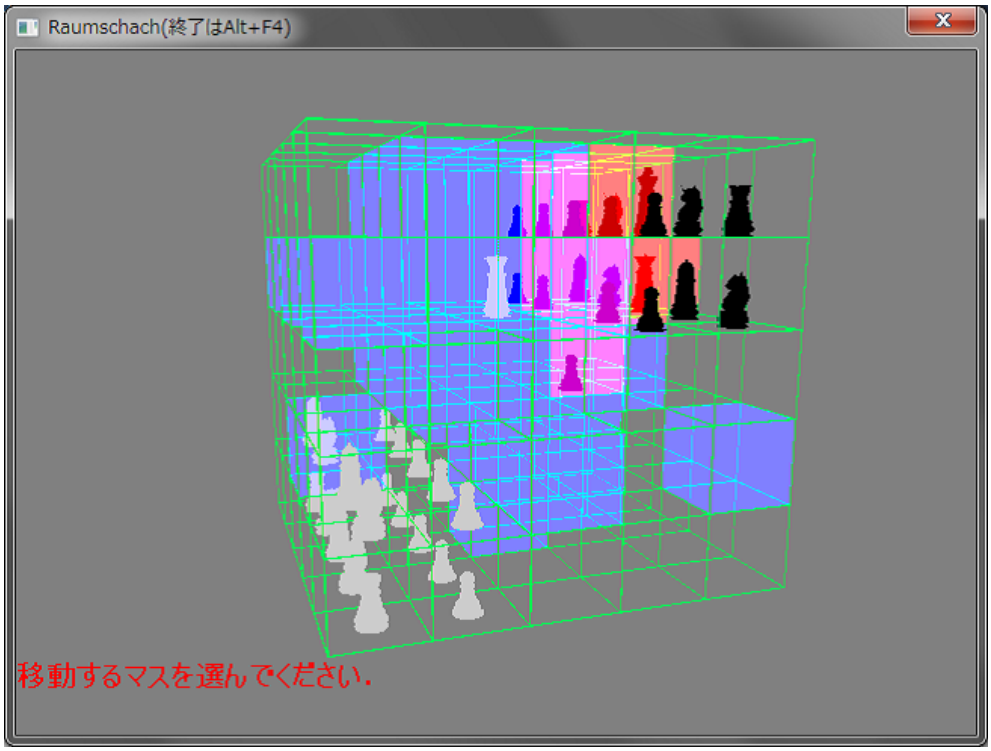

図 6　駒選択中の画面

終了画面で Enter キーを入力するとゲームは起動時の状態に戻る。終了はウィンドウの閉じるボタンか、Alt+F4 キーによって行う。終了するとインスタンスを全て破棄し、ウィンドウプロシージャの登録を解除。マウスカーソルを再度表示してアプリケーションは完全に終了する。

4　結論・今後の課題

本研究では、3D グラフィックを用いたラオムシャッハアプリケーションを開発した。本アプリケーションにより 2 人のプレイヤがラオムシャッハで対戦を行える。しかし、今回のアプリケーションではキーボード入

表 14　操作に用いるキー一覧

操作	キー					
	前	後	左	右	上	下
カーソル移動	↑	↓	←	→	/	\
カメラ移動	shift+↑	shift+↓	shift+←	shift+→	shift+/	shift+\
駒、移動先の選択	Enter					
ポーンの昇格先の選択	Q, N, B, R, U					
アプリケーションの終了	Alt + F4					

力を利用して操作するのだが、あまり操作性がいいとは言えず、改善するならば最低限マウス操作可能にする必要がある。また、仮にマウス操作可能にしたとしても操作性にそれほど劇的な変化は望めないだろう。したがって、今後増加するであろう 3D コンテンツをユーザが快適に操作するためには、3D コンテンツに適した革新的なユーザインタフェースが必要になると考えられる。例としてはマイクロソフト社が提供する Kinect [6] などがある。これらも SDK が公開されているため、一般の開発環境で利用することが可能である。また三次元空間に配置された駒の位置関係を通常のディスプレイを見て理解するのは難しく、視覚インタフェースの進化も必要ではないかと考えられる。だがホログラムディスプレイ等の適当な機能を持つ機材はまだ一般に浸透しているとは言えず、時間のかかる課題であると考えられる。

謝辞

本報告書の制作にあたり、数えきれないほどの御指導、御助言など大変尽力していただき、石水隆助教には誠に感謝申し上げます。本当にお世話になりました。ありがとうございます。

参考文献

[1] デスクトップ PC 向け AMD アクセラレーテッド・プロセッサー. AMD. http://www.amd.com/jp/products/desktop/apu/mainstream/Pages/mainstream.aspx

[2] インテル HD グラフィックス. Intel. http://www.intel.com/jp/technology/graphics/intelhd.htm

[3] 大槻有一郎. 15 歳からはじめる DirectX 9 3D ゲームプログラミング教室 C++ 編. 株式会社ラトルズ, 2007.

[4] A.S.M.Dickins. Guide to Fairy Chess. Dover Publications Inc, 1971.

[5] DirectX デベロッパー センター. Microsoft. http://msdn.microsoft.com/ja-jp/directx/default.

[6] Kinect for Windows SDK. Microsoft. http://www.microsoft.com/en-us/kinectforwindows/.

[7] Ray Edward Bornert II, The Game Of Three Dimensional Eight Level Chess, http://www.hixoxih.com/games/chess/3D8L.htm

[8] Dan Beyer, 3D CHESS, 2006. http://thehinge.net/3dchess/

[9] Ferdinand Maack, and Robert Price, Raumschach, 2001. http://zillionsofgames.com/cgi-bin/zilligames/submissions.cgi?do=show&id=708

付録 A　Windows アプリケーションのスケルトンプログラムのソースコード

```
#include<Windows.h>
#include<tchar.h>

/**
 * ウィンドウプロシージャ、OS からのメッセージに合わせ処理を行う
 * @param hWnd:ウィンドウのハンドル
 * @param msg:メッセージの識別子
 * @param wParam:メッセージの最初のパラメータ
 * @param lParam:メッセージの二番目のパラメータ
 * @return LRESULT:TRUE=1,FALSE=0,S_OK=0,S_FALSE=1
 */
LRESULT WINAPI MsgProc(HWND hWnd,UINT msg,WPARAM wParam,LPARAM lParam){
   //OS からメッセージを受け取り、それに合わせて処理を実行
   switch(msg){
   //ウィンドウが破棄された時
   case WM_DESTROY:
      //WM_QUIT(プログラムの終了) を送出
      PostQuitMessage(0);
      return 0;
   }
   //それら以外のメッセージが来たときはシステムの既定の処理を返す
   return DefWindowProc(hWnd,msg,wParam,lParam);
}

/**
 * WinMain 関数
 * @param hInst:現在のインスタンスのハンドル
 * @param hPrevInst:以前のインスタンスのハンドル
 * @param lpCmd:コマンドライン
 * @param nCmd:表示状態
 * @return int:終了時のステータス
 */
int WINAPI WinMain(HINSTANCE hInst,HINSTANCE hPInst,LPSTR lpCmd,int nCmd){
   //ウィンドウクラスの作成
   WNDCLASSEX wc = {sizeof(WNDCLASSEX),CS_CLASSDC,MsgProc,0L,0L,hInst,
```

```
        NULL,NULL,NULL,NULL,_T("My Window"),NULL};
    //ウィンドウクラスの登録
    RegisterClassEx(&wc);

    //ウィンドウの作成
    HWND hWnd = CreateWindow(_T("My Window"),_T("ウィンドウ名"),
        WS_OVERLAPPEDWINDOW,100,100,300,300,NULL,NULL,hInst,NULL);
    //ウィンドウの表示
    ShowWindow(hWnd,SW_SHOWDEFAULT);

    //メッセージを宣言し初期化
    MSG msg;
    ZeroMemory(&msg,sizeof(msg));
    //メッセージループここから
    //プログラムの終了メッセージを受け取るまで
    while(msg.message!=WM_QUIT){
        //メッセージを受け取る
        if(PeekMessage(&msg,NULL,0U,0U,PM_REMOVE)){
            //WM_KEYDOWN メッセージから WM_CHAR メッセージを作成
            TranslateMessage(&msg);
            //ウィンドウプロシージャにメッセージを転送
            DispatchMessage(&msg);
        }
    }
    //メッセージループここまで

    //ウィンドウクラスの登録解除
    UnregisterClass(_T("My Window"),hInst);
    return 0;
}
```

付録 B　Raumschach アプリケーションのソースコード

B.1　my3dlib.h

```
#include<Windows.h>
#include<stdio.h>
#include<MMSystem.h>
#include<d3dx9.h>
#include<tchar.h>
```

```
#include<dinput.h>

//X ファイル管理用構造体ここから
struct Model{
   //メッシュ
   LPD3DXMESH   pmesh;
   //マテリアルの配列
   D3DMATERIAL9*   pmaterials;
   //テクスチャの配列
   LPDIRECT3DTEXTURE9*   ptextures;
   //マテリアルの数
   DWORD   nummaterials;
   //データが入っているか示すフラグ
   BOOL   used;
};
//X ファイル管理用構造体ここまで

//グローバル変数（宣言）ここから
//IDirect3D9 へのポインタ
extern LPDIRECT3D9   g_pD3D;
//IDirect3DDevice9 へのポインタ
extern LPDIRECT3DDEVICE9   g_pd3dDevice;
//ウィンドウのアスペクト比（幅÷横）
extern float   g_aspect;
//モデルデータ管理配列
extern Model   g_models[];
//フォントデータ管理配列
extern LPD3DXFONT   g_pxfonts[];
//テキストスプライトのポインタ
extern LPD3DXSPRITE   g_ptextsprite;

//ラオムシャッハのチェス盤（メモリ確保のため 15）
extern int   board[15][15][15];
//移動・攻撃判定用のチェス盤
extern int   tmp[15][15][15];
//勝者
extern int   winner;
//現在のプレイヤーの色
extern int   currentTurn;
//現在のゲームの状態
```

```
extern int   currentMode;
//チェックメイトであるかどうかの判定子
extern bool   checkmateCall;
//グローバル変数（宣言）ここまで

//関数プロトタイプ宣言ここから
//D3D 初期化（ウィンドウモード）
HRESULT InitD3DWindow(LPCTSTR wintitle,int w,int h);
//モデルデータの読みこみ
int LoadModel(LPCTSTR filename);
//モデルの描画
void RenderModel(int idx);
//キーボード入力の検知
const char *GetKeyState();
//タイマーをセット
void setTimer(int idx,DWORD time);
//タイマーがセットした時間に到達すると TRUE
BOOL isTimerGoal(int idx);
//タイマーの経過時間
DWORD getPassedTime(int idx);
//フォントファイルの読み込み
int CreateGameFont(LPCTSTR fontname,int height,UINT weight);

//ラオムシャッハの初期配置を行う
void reset();
//ある駒の移動・攻撃可能範囲を tmp に格納
void moveCheck(int x,int y,int z);
//勝者を調べる
bool winCheck();
//チェックされるかどうか
bool checkCheck(int kcolor);
//ステールメイトでないか
bool stalemateCheck(int color);
//プロモーションが起こるか
bool promotionCheck(int color);
//関数プロトタイプ宣言ここまで
```

B.2 my3dlib.cpp

```
#include"my3dlib.h"

//グローバル変数ここから
//IDirect3D9 へのポインタ
LPDIRECT3D9      g_pD3D = NULL;
//IDirect3DDevice9 へのポインタ
LPDIRECT3DDEVICE9   g_pd3dDevice = NULL;
//ウィンドウのアスペクト比（幅÷横）
float         g_aspect = 1.0f;
//読み込む X ファイルの最大個数
const int      MAXMODEL = 64;
//モデルデータ管理配列
Model         g_models[MAXMODEL];
//IDirectInput8 へのポインタ
LPDIRECTINPUT8      g_pDI = NULL;
//IDirectInputDevice8 へのポインタ
LPDIRECTINPUTDEVICE8   g_pDIDevice = NULL;
//受け取ったキーボードの状態を格納
char         g_keys[256];
//タイマーの最大数
const int      MAXTIMER = 64;
//タイマー管理用配列
DWORD         g_goaltimes[MAXTIMER];
//最大フォント数
const int      MAXFONT = 64;
//フォントデータ管理配列
LPD3DXFONT      g_pxfonts[MAXFONT];
//テキストスプライトのポインタ
LPD3DXSPRITE      g_ptextsprite = NULL;
//グローバル変数ここまで

/**
 * タイマーをセットする関数
 * @param idx:インデックス
 * @param time:時間
 * @return void
 */
```

```
void setTimer(int idx,DWORD time){
   //限界要素数を超えない
   if(idx>MAXTIMER)return;
   g_goaltimes[idx] = timeGetTime() + time;
}
/**
 * タイマーが設定した時間を超えたか確認する関数
 * @param idx:インデックス
 * @return BOOL:タイマーが設定時間を超えているならば TRUE=1
 */
BOOL isTimerGoal(int idx){
   if(g_goaltimes[idx]>timeGetTime())return FALSE;
   return TRUE;
}
/**
 * タイマーの経過時間を調べる関数
 * @param idx:インデックス
 * @return DWORD:時間（ミリ秒）
 */
DWORD getPassedTime(int idx){
   return timeGetTime() - g_goaltimes[idx];
}

/**
 * リソース（各種インスタンス）の解放を行う関数
 * @param void
 * @return void
 */
void CleanupD3D(){
   //全てのフォントインスタンスを解放
   for(int i=0;i<MAXFONT;i++){
      if(g_pxfonts[i])g_pxfonts[i]->Release();
   }
   //テキストスプライトの解放
   if(g_ptextsprite)g_ptextsprite->Release();

   //全てのメッシュ（モデル・マテリアル・テクスチャ）の解放
   for(int i=0;i<MAXMODEL;i++){
      if(g_models[i].used != TRUE){
         if(g_models[i].pmaterials != NULL){
```

```
            delete[] g_models[i].pmaterials;
        }
        if(g_models[i].ptextures != NULL){
            for(DWORD j=0;j<g_models[i].nummaterials;j++){
                g_models[i].ptextures[j]->Release();
            }
            delete[] g_models[i].ptextures;
        }
        if(g_models[i].pmesh != NULL){
            g_models[i].pmesh->Release();
        }
    }
  }
  //DirectInputDevice の解放
  if(g_pDIDevice != NULL){
    g_pDIDevice->Unacquire();
    g_pDIDevice->Release();
  }
  if(g_pDI != NULL)g_pDI->Release();
  //DirectX デバイスの解放
  if(g_pd3dDevice != NULL)g_pd3dDevice->Release();
  if(g_pD3D != NULL)g_pD3D->Release();
}

/**
 * ウィンドウプロシージャ、OS からのメッセージに合わせ処理を行う
 * @param hWnd:ウィンドウのハンドル
 * @param msg:メッセージの識別子
 * @param wParam:メッセージの最初のパラメータ
 * @param lParam:メッセージの二番目のパラメータ
 * @return LRESULT:TRUE=1,FALSE=0,S_OK=0,S_FALSE=1
 */
LRESULT WINAPI MsgProc(HWND hWnd,UINT msg,WPARAM wParam,LPARAM lParam){
  //OS からメッセージを受け取り、それに合わせて処理を実行
  switch(msg){
  //ウィンドウが破棄された時
  case WM_DESTROY:
    //リソースを解放
    CleanupD3D();
    //WM_QUIT(プログラムの終了) を送出
```

```
        PostQuitMessage(0);
        return 0;
    }
    //それら以外のメッセージが来たときはシステムの既定の処理を返す
    return DefWindowProc(hWnd,msg,wParam,lParam);
}

/**
 * 各種インスタンスの作成初期化・ウィンドウクラスの作成登録
 * @param void
 * @return void
 */
HRESULT InitD3DWindow(LPCTSTR wintitle,int w,int h){
    //モデルとフォントの配列を初期化
    ZeroMemory(&g_models,sizeof(Model)*MAXMODEL);
    ZeroMemory(&g_pxfonts,sizeof(LPD3DXFONT)*MAXFONT);
    //ウィンドウクラス作成
    WNDCLASSEX wc = {sizeof(WNDCLASSEX),CS_CLASSDC,MsgProc,0L,0L,
        GetModuleHandle(NULL),NULL,NULL,NULL,NULL,
        _T("D3D Window Class"),NULL};
    //ウィンドウクラスの登録
    RegisterClassEx(&wc);

    //ウィンドウ作成
    HWND hWnd = CreateWindow(_T("D3D Window Class"),wintitle,
        WS_OVERLAPPED | WS_SYSMENU,100,100,w,h,
        NULL,NULL,wc.hInstance,NULL);
    //ビューポートアスペクトの記録
    g_aspect = (float)w/(float)h;

    //D3D9の作成
    if(NULL == (g_pD3D = Direct3DCreate9(D3D_SDK_VERSION)))return E_FAIL;

    //D3Dデバイスの作成（可能ならばハードウェアT&Lを使用）
    D3DPRESENT_PARAMETERS d3dpp;
    ZeroMemory(&d3dpp,sizeof(d3dpp));
    //ウィンドウモードで
    d3dpp.Windowed = TRUE;
    //画面切り替えを瞬時に
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
```

```
    //バックバッファの型はシステムデフォルト
    d3dpp.BackBufferFormat = D3DFMT_UNKNOWN;
    //Z バッファを有効に
    d3dpp.EnableAutoDepthStencil = TRUE;
    //Z バッファは 16 ビット深度
    d3dpp.AutoDepthStencilFormat = D3DFMT_D16;

    //ハードウェア (GPU) サポートがあるなら受ける、無理ならソフトウェアでデバイスのインスタンスを作
成
    if(FAILED(g_pD3D->CreateDevice(D3DADAPTER_DEFAULT,D3DDEVTYPE_HAL,
       hWnd,D3DCREATE_HARDWARE_VERTEXPROCESSING,
       &d3dpp,&g_pd3dDevice))){
          if(FAILED(g_pD3D->CreateDevice(D3DADAPTER_DEFAULT,D3DDEVTYPE_HAL,
             hWnd,D3DCREATE_SOFTWARE_VERTEXPROCESSING,
             &d3dpp,&g_pd3dDevice))){
             return E_FAIL;
          }
    }

    //Z バッファをオン
    g_pd3dDevice->SetRenderState(D3DRS_ZENABLE,TRUE);
    //ウィンドウの表示
    ShowWindow(hWnd,SW_SHOWDEFAULT);

    //DirectInput の初期化
    if(FAILED(DirectInput8Create(wc.hInstance,DIRECTINPUT_VERSION,IID_IDirectInput8,
       (void**)&g_pDI,NULL)))return E_FAIL;
    if(FAILED(g_pDI->CreateDevice(GUID_SysKeyboard,&g_pDIDevice,NULL)))return E_FAIL;
    g_pDIDevice->SetDataFormat(&c_dfDIKeyboard);
    g_pDIDevice->SetCooperativeLevel(hWnd,DISCL_FOREGROUND | DISCL_NONEXCLUSIVE);

    //テキストスプライトの作成
    if(FAILED(D3DXCreateSprite(g_pd3dDevice,&g_ptextsprite)))return E_FAIL;

    return S_OK;
}

/**
 * X ファイルからモデルデータを読み込む関数
 * @param filename:ファイル名（同フォルダ内）
```

```
 * @return int:インデックス
 */
int LoadModel(LPCTSTR filename){
   //未使用の要素を探す（used が FALSE）
   int idx;
   for(idx=0;idx<MAXMODEL;idx++){
      if(g_models[idx].used == FALSE)break;
   }
   if(idx>=MAXMODEL)return -1;

   //一時記憶バッファ
   LPD3DXBUFFER pD3DXMtrlBuffer;
   //X ファイルの読み込み
   if(FAILED(D3DXLoadMeshFromX(filename,D3DXMESH_SYSTEMMEM,g_pd3dDevice,NULL,
      &pD3DXMtrlBuffer,NULL,&g_models[idx].nummaterials,
      &g_models[idx].pmesh))){
         MessageBox(NULL,_T("X ファイルが見つかりません"),_T("3D Lib"),MB_OK);
         return -1;
   }

   //マテリアルとテクスチャ記録用配列の確保
   D3DXMATERIAL* d3dxMaterials = (D3DXMATERIAL*)pD3DXMtrlBuffer->GetBufferPointer();
   int num = g_models[idx].nummaterials;
   g_models[idx].pmaterials = new D3DMATERIAL9[num];
   if(g_models[idx].pmaterials == NULL)return -1;
   g_models[idx].ptextures = new LPDIRECT3DTEXTURE9[num];
   if(g_models[idx].ptextures == NULL)return -1;

   for(int i=0;i<num;i++){
      //マテリアルのコピー
      g_models[idx].pmaterials[i]=d3dxMaterials[i].MatD3D;
      //アンビエント色の設定
      g_models[idx].pmaterials[i].Ambient=g_models[idx].pmaterials[i].Diffuse;
      //テクスチャの読み込み
      g_models[idx].ptextures[i]=NULL;
      if(d3dxMaterials[i].pTextureFilename != NULL &&
         lstrlenA(d3dxMaterials[i].pTextureFilename) > 0){
         if(FAILED(D3DXCreateTextureFromFileA(g_pd3dDevice,d3dxMaterials[i].pTextureFilename,
            &g_models[idx].ptextures[i]))){
            MessageBox(NULL,_T("テクスチャが見つかりません"),_T("3D Lib"),MB_OK);
```

```
            return -1;
         }
      }
   }

   pD3DXMtrlBuffer->Release();
   g_models[idx].used = TRUE;
   return idx;
}

/**
 * モデルデータの描画を行う関数
 * @param idx; インデックス
 * @return void
 */
void RenderModel(int idx){
   if(g_models[idx].used == FALSE)return;

   for(DWORD i=0;i<g_models[idx].nummaterials;i++){
      g_pd3dDevice->SetMaterial(&g_models[idx].pmaterials[i]);
      g_pd3dDevice->SetTexture(0,g_models[idx].ptextures[i]);
      g_models[idx].pmesh->DrawSubset(i);
   }
}

/**
 * キーボードの状態を keys に格納する関数
 * @param void
 * @return char:キーボード状態
 */
const char *GetKeyState(){
   HRESULT hr = g_pDIDevice->Acquire();
   if((hr==DI_OK)||(hr==S_FALSE)){
      g_pDIDevice->GetDeviceState(sizeof(g_keys),&g_keys);
      return g_keys;
   }
   return NULL;
}

/**
```

```
 * フォントファイルからフォントデータを作成する関数
 * @param fontname:フォントファイル名（「MS ゴシック」等）
 * @param height:文字の高さ（大きさ）
 * @param weight:文字の太さ
 * @return int:インデックス
 */
int CreateGameFont(LPCTSTR fontname,int height,UINT weight){
   //空いている要素を探す
   int idx;
   for(idx=0;idx<MAXFONT;idx++){
      if(g_pxfonts[idx]==NULL)break;
   }
   if(idx>=MAXFONT)return -1;

   //フォントを作成する
   HRESULT hr = D3DXCreateFont(g_pd3dDevice,-height,0,weight,1,FALSE,
      DEFAULT_CHARSET,OUT_DEFAULT_PRECIS,DEFAULT_QUALITY,
      DEFAULT_PITCH | FF_DONTCARE,fontname,&g_pxfonts[idx]);
   if(FAILED(hr))return -1;

   return idx;
}
```

B.3 game.cpp

```
#include"my3dlib.h"

int board[15][15][15];   //ゲーム盤
int tmp[15][15][15];   //コマの移動範囲判定用
int collision[26];   //コマ同士の衝突を判断する配列

enum{   //マスの状態
ATT=-2,   //攻撃可能
MOV,   //移動可能
NONE,   //0 コマなし
W_K,   //1 白キング K
W_Q,   //2 白クイーン Q
W_U,   //3 白ユニコーン U
W_R,   //4 白ルーク R
```

```
W_B,   //5 白ビショップ B
W_N,   //6 白ナイト N
W_P,   //7 白ポーン P
B_K,   //8 黒キング K
B_Q,   //9 黒クイーン Q
B_U,   //10 黒ユニコーン U
B_R,   //11 黒ルーク R
B_B,   //12 黒ビショップ B
B_N,   //13 黒ナイト N
B_P   //14 黒ポーン P
};


/**
 * ラオムシャッハの初期配置を行う関数
 * @param void
 * @return void
 */
void reset(){
   for(int i=0;i<5;i++){
      for(int j=0;j<5;j++){
         for(int k=0;k<5;k++){
            board[i][j][k]=0;
         }
      }
   }

   //黒初期配置
   board[4][4][2]=B_K;
   board[3][4][2]=B_Q;
   board[4][4][1]=board[4][4][3]=B_N;
   board[4][4][0]=board[4][4][4]=B_R;
   board[3][4][0]=board[3][4][3]=B_B;
   board[3][4][1]=board[3][4][4]=B_U;
   for(int i=0;i<5;i++){
      board[4][3][i]=board[3][3][i]=B_P;
   }
   //白初期配置
   board[0][0][2]=W_K;
   board[1][0][2]=W_Q;
```

```
   board[0][0][1]=board[0][0][3]=W_N;
   board[0][0][0]=board[0][0][4]=W_R;
   board[1][0][0]=board[1][0][3]=W_B;
   board[1][0][1]=board[1][0][4]=W_U;
   for(int i=0;i<5;i++){
      board[0][1][i]=board[1][1][i]=W_P;
   }
}
/**
 * 指定された駒の移動範囲を確認し tmp に格納する関数
 * @param x,y,z:駒の座標
 * @return void
 */
void moveCheck(int x,int y,int z){
   int t=board[z][y][x];
   for(int i=0;i<5;i++){
      for(int j=0;j<5;j++){
         for(int k=0;k<5;k++){
            tmp[k][j][i]=board[k][j][i];
         }
      }
   }

   for(int i=0;i<26;i++){
      collision[i]=0;
   }
   switch(tmp[z][y][x]){
      //コマの色が白（1〜7 の場合）
      case W_K:{
         for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
               for(int k=0;k<3;k++){
                  if(tmp[z-1+i][y-1+j][x-1+k]==NONE){
                     tmp[z-1+i][y-1+j][x-1+k]=MOV;
                  }else if(8<=tmp[z-1+i][y-1+j][x-1+k]&&
                     tmp[z-1+i][y-1+j][x-1+k]<=14){
                     tmp[z-1+i][y-1+j][x-1+k]=ATT;
                  }
               }
            }
```

```
   }
   break;
    }
case W_Q:{
   for(int i=1;i<5;i++){
      //ROOK
      //二次元面
      if(!collision[0]){if(x+i<5){
         if(tmp[z][y][x+i]==NONE){tmp[z][y][x+i]=-1;}
      else if(8<=tmp[z][y][x+i]&&tmp[z][y][x+i]<=14){
         tmp[z][y][x+i]=-2;collision[0]++;}
      else collision[0]++;}}
      if(!collision[1]){if(x-i>=0){
         if(tmp[z][y][x-i]==NONE){tmp[z][y][x-i]=-1;}
      else if(8<=tmp[z][y][x-i]&&tmp[z][y][x-i]<=14){
         tmp[z][y][x-i]=-2;collision[1]++;}
      else collision[1]++;}}
      if(!collision[2]){if(y+i<5){
         if(tmp[z][y+i][x]==NONE){tmp[z][y+i][x]=-1;}
      else if(8<=tmp[z][y+i][x]&&tmp[z][y+i][x]<=14){
         tmp[z][y+i][x]=-2;collision[2]++;}
      else collision[2]++;}}
      if(!collision[3]){if(y-i>=0){
         if(tmp[z][y-i][x]==NONE){tmp[z][y-i][x]=-1;}
      else if(8<=tmp[z][y-i][x]&&tmp[z][y-i][x]<=14){
         tmp[z][y-i][x]=-2;collision[3]++;}
      else collision[3]++;}}
      //三次元面
      if(!collision[4]){if(z+i<5){
         if(tmp[z+i][y][x]==NONE){tmp[z+i][y][x]=-1;}
      else if(8<=tmp[z+i][y][x]&&tmp[z+i][y][x]<=14){
         tmp[z+i][y][x]=-2;collision[4]++;}
      else collision[4]++;}}
      if(!collision[5]){if(z-i>=0){
         if(tmp[z-i][y][x]==NONE){tmp[z-i][y][x]=-1;}
      else if(8<=tmp[z-i][y][x]&&tmp[z-i][y][x]<=14){
         tmp[z-i][y][x]=-2;collision[5]++;}
      else collision[5]++;}}
      //BISHOP
      //二次元面
```

```
if(!collision[6]){if(x+i<5 && y+i<5){
   if(tmp[z][y+i][x+i]==NONE){tmp[z][y+i][x+i]=-1;}
else if(8<=tmp[z][y+i][x+i]&&tmp[z][y+i][x+i]<=14){
   tmp[z][y+i][x+i]=-2;collision[6]++;}
else collision[6]++;}}
if(!collision[7]){if(x+i<5 && y-i>=0){
   if(tmp[z][y-i][x+i]==NONE){tmp[z][y-i][x+i]=-1;}
else if(8<=tmp[z][y-i][x+i]&&tmp[z][y-i][x+i]<=14){
   tmp[z][y-i][x+i]=-2;collision[7]++;}
else collision[7]++;}}
if(!collision[8]){if(x-i>=0 && y+i<5){
   if(tmp[z][y+i][x-i]==NONE){tmp[z][y+i][x-i]=-1;}
else if(8<=tmp[z][y+i][x-i]&&tmp[z][y+i][x-i]<=14){
   tmp[z][y+i][x-i]=-2;collision[8]++;}
else collision[8]++;}}
if(!collision[9]){if(x-i>=0 && y-i>=0){
   if(tmp[z][y-i][x-i]==NONE){tmp[z][y-i][x-i]=-1;}
else if(8<=tmp[z][y-i][x-i]&&tmp[z][y-i][x-i]<=14){
   tmp[z][y-i][x-i]=-2;collision[9]++;}
else collision[9]++;}}
//三次元面上方向
if(!collision[10]){if(z+i<5 && x+i<5){
   if(tmp[z+i][y][x+i]==NONE){tmp[z+i][y][x+i]=-1;}
else if(8<=tmp[z+i][y][x+i]&&tmp[z+i][y][x+i]<=14){
   tmp[z+i][y][x+i]=-2;collision[10]++;}
else collision[10]++;}}
if(!collision[11]){if(z+i<5 && x-i>=0){
   if(tmp[z+i][y][x-i]==NONE){tmp[z+i][y][x-i]=-1;}
else if(8<=tmp[z+i][y][x-i]&&tmp[z+i][y][x-i]<=14){
   tmp[z+i][y][x-i]=-2;collision[11]++;}
else collision[11]++;}}
if(!collision[12]){if(z+i<5 && y+i<5){
   if(tmp[z+i][y+i][x]==NONE){tmp[z+i][y+i][x]=-1;}
else if(8<=tmp[z+i][y+i][x]&&tmp[z+i][y+i][x]<=14){
   tmp[z+i][y+i][x]=-2;collision[12]++;}
else collision[12]++;}}
if(!collision[13]){if(z+i<5 && y-i>=0){
   if(tmp[z+i][y-i][x]==NONE){tmp[z+i][y-i][x]=-1;}
else if(8<=tmp[z+i][y-i][x]&&tmp[z+i][y-i][x]<=14){
   tmp[z+i][y-i][x]=-2;collision[13]++;}
```

```
else collision[13]++;}}
//三次元面下方向
if(!collision[14]){if(z-i>=0 && x+i<5){
   if(tmp[z-i][y][x+i]==NONE){tmp[z-i][y][x+i]=-1;}
else if(8<=tmp[z-i][y][x+i]&&tmp[z-i][y][x+i]<=14){
   tmp[z-i][y][x+i]=-2;collision[14]++;}
else collision[14]++;}}
if(!collision[15]){if(z-i>=0 && x-i>=0){
   if(tmp[z-i][y][x-i]==NONE){tmp[z-i][y][x-i]=-1;}
else if(8<=tmp[z-i][y][x-i]&&tmp[z-i][y][x-i]<=14){
   tmp[z-i][y][x-i]=-2;collision[15]++;}
else collision[15]++;}}
if(!collision[16]){if(z-i>=0 && y+i<5){
   if(tmp[z-i][y+i][x]==NONE){tmp[z-i][y+i][x]=-1;}
else if(8<=tmp[z-i][y+i][x]&&tmp[z-i][y+i][x]<=14){
   tmp[z-i][y+i][x]=-2;collision[16]++;}
else collision[16]++;}}
if(!collision[17]){if(z-i>=0 && y-i>=0){
   if(tmp[z-i][y-i][x]==NONE){tmp[z-i][y-i][x]=-1;}
else if(8<=tmp[z-i][y-i][x]&&tmp[z-i][y-i][x]<=14){
   tmp[z-i][y-i][x]=-2;collision[17]++;}
else collision[17]++;}}
//UNICORN
//三次元面上方向
if(!collision[18]){if(z+i<5 && x+i<5 && y+i<5){
   if(tmp[z+i][y+i][x+i]==NONE){tmp[z+i][y+i][x+i]=-1;}
else if(8<=tmp[z+i][y+i][x+i]&&tmp[z+i][y+i][x+i]<=14){
   tmp[z+i][y+i][x+i]=-2;collision[18]++;}
else collision[18]++;}}
if(!collision[19]){if(z+i<5 && x+i<5 && y-i>=0){
   if(tmp[z+i][y-i][x+i]==NONE){tmp[z+i][y-i][x+i]=-1;}
else if(8<=tmp[z+i][y-i][x+i]&&tmp[z+i][y-i][x+i]<=14){
   tmp[z+i][y-i][x+i]=-2;collision[19]++;}
else collision[19]++;}}
if(!collision[20]){if(z+i<5 && x-i>=0 && y+i<5){
   if(tmp[z+i][y+i][x-i]==NONE){tmp[z+i][y+i][x-i]=-1;}
else if(8<=tmp[z+i][y+i][x-i]&&tmp[z+i][y+i][x-i]<=14){
   tmp[z+i][y+i][x-i]=-2;collision[20]++;}
else collision[20]++;}}
if(!collision[21]){if(z+i<5 && x-i>=0 && y-i>=0){
```

```
            if(tmp[z+i][y-i][x-i]==NONE){tmp[z+i][y-i][x-i]=-1;}
         else if(8<=tmp[z+i][y-i][x-i]&&tmp[z+i][y-i][x-i]<=14){
            tmp[z+i][y-i][x-i]=-2;collision[21]++;}
         else collision[21]++;}}
         //三次元面下方向
         if(!collision[22]){if(z-i>=0 && x+i<5 && y+i<5){
            if(tmp[z-i][y+i][x+i]==NONE){tmp[z-i][y+i][x+i]=-1;}
         else if(8<=tmp[z-i][y+i][x+i]&&tmp[z-i][y+i][x+i]<=14){
            tmp[z-i][y+i][x+i]=-2;collision[22]++;}
         else collision[22]++;}}
         if(!collision[23]){if(z-i>=0 && x+i<5 && y-i>=0){
            if(tmp[z-i][y-i][x+i]==NONE){tmp[z-i][y-i][x+i]=-1;}
         else if(8<=tmp[z-i][y-i][x+i]&&tmp[z-i][y-i][x+i]<=14){
            tmp[z-i][y-i][x+i]=-2;collision[23]++;}
         else collision[23]++;}}
         if(!collision[24]){if(z-i>=0 && x-i>=0 && y+i<5){
            if(tmp[z-i][y+i][x-i]==NONE){tmp[z-i][y+i][x-i]=-1;}
         else if(8<=tmp[z-i][y+i][x-i]&&tmp[z-i][y+i][x-i]<=14){
            tmp[z-i][y+i][x-i]=-2;collision[24]++;}
         else collision[24]++;}}
         if(!collision[25]){if(z-i>=0 && x-i>=0 && y-i>=0){
            if(tmp[z-i][y-i][x-i]==NONE){tmp[z-i][y-i][x-i]=-1;}
         else if(8<=tmp[z-i][y-i][x-i]&&tmp[z-i][y-i][x-i]<=14){
            tmp[z-i][y-i][x-i]=-2;collision[25]++;}
         else collision[25]++;}}
      }
      break;
           }
   case W_N:{
      if(y+1<5 && x+2<5){
         if(tmp[z][y+1][x+2]==NONE)tmp[z][y+1][x+2]=-1;
      else if(8<=tmp[z][y+1][x+2] && tmp[z][y+1][x+2]<=14)
         tmp[z][y+1][x+2]=-2;}
      if(y+1<5 && x-2>=0){
         if(tmp[z][y+1][x-2]==NONE)tmp[z][y+1][x-2]=-1;
      else if(8<=tmp[z][y+1][x-2] && tmp[z][y+1][x-2]<=14)
         tmp[z][y+1][x-2]=-2;}
      if(y-1>=0 && x+2<5){
         if(tmp[z][y-1][x+2]==NONE)tmp[z][y-1][x+2]=-1;
      else if(8<=tmp[z][y-1][x+2] && tmp[z][y-1][x+2]<=14)
```

```
   tmp[z][y-1][x+2]=-2;}
if(y-1>=0 && x-2>=0){
   if(tmp[z][y-1][x-2]==NONE)tmp[z][y-1][x-2]=-1;
else if(8<=tmp[z][y-1][x-2] && tmp[z][y-1][x-2]<=14)
   tmp[z][y-1][x-2]=-2;}
if(y+2<5 && x+1<5){
   if(tmp[z][y+2][x+1]==NONE)tmp[z][y+2][x+1]=-1;
else if(8<=tmp[z][y+2][x+1] && tmp[z][y+2][x+1]<=14)
   tmp[z][y+2][x+1]=-2;}
if(y+2<5 && x-1>=0){
   if(tmp[z][y+2][x-1]==NONE)tmp[z][y+2][x-1]=-1;
else if(8<=tmp[z][y+2][x-1] && tmp[z][y+2][x-1]<=14)
   tmp[z][y+2][x-1]=-2;}
if(y-2>=0 && x+1<5){
   if(tmp[z][y-2][x+1]==NONE)tmp[z][y-2][x+1]=-1;
else if(8<=tmp[z][y-2][x+1] && tmp[z][y-2][x+1]<=14)
   tmp[z][y-2][x+1]=-2;}
if(y-2>=0 && x-1>=0){
   if(tmp[z][y-2][x-1]==NONE)tmp[z][y-2][x-1]=-1;
else if(8<=tmp[z][y-2][x-1] && tmp[z][y-2][x-1]<=14)
   tmp[z][y-2][x-1]=-2;}

if(z+1<5 && x+2<5){if(tmp[z+1][y][x+2]==NONE)
   tmp[z+1][y][x+2]=-1;
else if(8<=tmp[z+1][y][x+2] && tmp[z+1][y][x+2]<=14)
   tmp[z+1][y][x+2]=-2;}
if(z+1<5 && x-2>=0){if(tmp[z+1][y][x-2]==NONE)
   tmp[z+1][y][x-2]=-1;
else if(8<=tmp[z+1][y][x-2] && tmp[z+1][y][x-2]<=14)
   tmp[z+1][y][x-2]=-2;}
if(z-1>=0 && x+2<5){if(tmp[z-1][y][x+2]==NONE)
   tmp[z-1][y][x+2]=-1;
else if(8<=tmp[z-1][y][x+2] && tmp[z-1][y][x+2]<=14)
   tmp[z-1][y][x+2]=-2;}
if(z-1>=0 && x-2>=0){if(tmp[z-1][y][x-2]==NONE)
   tmp[z-1][y][x-2]=-1;
else if(8<=tmp[z-1][y][x-2] && tmp[z-1][y][x-2]<=14)
   tmp[z-1][y][x-2]=-2;}
if(z+1<5 && y+2<5){if(tmp[z+1][y+2][x]==NONE)
   tmp[z+1][y+2][x]=-1;
```

```
else if(8<=tmp[z+1][y+2][x] && tmp[z+1][y+2][x]<=14)
   tmp[z+1][y+2][x]=-2;}
if(z+1<5 && y-2>=0){if(tmp[z+1][y-2][x]==NONE)
   tmp[z+1][y-2][x]=-1;
else if(8<=tmp[z+1][y-2][x] && tmp[z+1][y-2][x]<=14)
   tmp[z+1][y-2][x]=-2;}
if(z-1>=0 && y+2<5){if(tmp[z-1][y+2][x]==NONE)
   tmp[z-1][y+2][x]=-1;
else if(8<=tmp[z-1][y+2][x] && tmp[z-1][y+2][x]<=14)
   tmp[z-1][y+2][x]=-2;}
if(z-1>=0 && y-2>=0){if(tmp[z-1][y-2][x]==NONE)
   tmp[z-1][y-2][x]=-1;
else if(8<=tmp[z-1][y-2][x] && tmp[z-1][y-2][x]<=14)
   tmp[z-1][y-2][x]=-2;}

if(z+2<5 && x+1<5){if(tmp[z+2][y][x+1]==NONE)
   tmp[z+2][y][x+1]=-1;
else if(8<=tmp[z+2][y][x+1] && tmp[z+2][y][x+1]<=14)
   tmp[z+2][y][x+1]=-2;}
if(z+2<5 && x-1>=0){if(tmp[z+2][y][x-1]==NONE)
   tmp[z+2][y][x-1]=-1;
else if(8<=tmp[z+2][y][x-1] && tmp[z+2][y][x-1]<=14)
   tmp[z+2][y][x-1]=-2;}
if(z-2>=0 && x+1<5){if(tmp[z-2][y][x+1]==NONE)
   tmp[z-2][y][x+1]=-1;
else if(8<=tmp[z-2][y][x+1] && tmp[z-2][y][x+1]<=14)
   tmp[z-2][y][x+1]=-2;}
if(z-2>=0 && x-1>=0){if(tmp[z-2][y][x-1]==NONE)
   tmp[z-2][y][x-1]=-1;
else if(8<=tmp[z-2][y][x-1] && tmp[z-2][y][x-1]<=14)
   tmp[z-2][y][x-1]=-2;}
if(z+2<5 && y+1<5){if(tmp[z+2][y+1][x]==NONE)
   tmp[z+2][y+1][x]=-1;
else if(8<=tmp[z+2][y+1][x] && tmp[z+2][y+1][x]<=14)
   tmp[z+2][y+1][x]=-2;}
if(z+2<5 && y-1>=0){if(tmp[z+2][y-1][x]==NONE)
   tmp[z+2][y-1][x]=-1;
else if(8<=tmp[z+2][y-1][x] && tmp[z+2][y-1][x]<=14)
   tmp[z+2][y-1][x]=-2;}
if(z-2>=0 && y+1<5){if(tmp[z-2][y+1][x]==NONE)
```

```
      tmp[z-2][y+1][x]=-1;
   else if(8<=tmp[z-2][y+1][x] && tmp[z-2][y+1][x]<=14)
      tmp[z-2][y+1][x]=-2;}
   if(z-2>=0 && y-1>=0){if(tmp[z-2][y-1][x]==NONE)
      tmp[z-2][y-1][x]=-1;
   else if(8<=tmp[z-2][y-1][x] && tmp[z-2][y-1][x]<=14)
      tmp[z-2][y-1][x]=-2;}
   break;
         }
case W_R:{
   for(int i=1;i<5;i++){
      //二次元面
      if(!collision[0]){if(x+i<5){
         if(tmp[z][y][x+i]==NONE){tmp[z][y][x+i]=-1;}
      else if(8<=tmp[z][y][x+i]&&tmp[z][y][x+i]<=14){
         tmp[z][y][x+i]=-2;collision[0]++;}
      else collision[0]++;}}
      if(!collision[1]){if(x-i>=0){
         if(tmp[z][y][x-i]==NONE){tmp[z][y][x-i]=-1;}
      else if(8<=tmp[z][y][x-i]&&tmp[z][y][x-i]<=14){
         tmp[z][y][x-i]=-2;collision[1]++;}
      else collision[1]++;}}
      if(!collision[2]){if(y+i<5){
         if(tmp[z][y+i][x]==NONE){tmp[z][y+i][x]=-1;}
      else if(8<=tmp[z][y+i][x]&&tmp[z][y+i][x]<=14){
         tmp[z][y+i][x]=-2;collision[2]++;}
      else collision[2]++;}}
      if(!collision[3]){if(y-i>=0){
         if(tmp[z][y-i][x]==NONE){tmp[z][y-i][x]=-1;}
      else if(8<=tmp[z][y-i][x]&&tmp[z][y-i][x]<=14){
         tmp[z][y-i][x]=-2;collision[3]++;}
      else collision[3]++;}}
      //三次元面
      if(!collision[4]){if(z+i<5){
         if(tmp[z+i][y][x]==NONE){tmp[z+i][y][x]=-1;}
      else if(8<=tmp[z+i][y][x]&&tmp[z+i][y][x]<=14){
         tmp[z+i][y][x]=-2;collision[4]++;}
      else collision[4]++;}}
      if(!collision[5]){if(z-i>=0){
         if(tmp[z-i][y][x]==NONE){tmp[z-i][y][x]=-1;}
```

```
      else if(8<=tmp[z-i][y][x]&&tmp[z-i][y][x]<=14){
         tmp[z-i][y][x]=-2;collision[5]++;}
      else collision[5]++;}}
   }
   break;
       }
case W_B:{
   for(int i=1;i<5;i++){
      //二次元面
      if(!collision[0]){if(x+i<5 && y+i<5){
         if(tmp[z][y+i][x+i]==NONE){tmp[z][y+i][x+i]=-1;}
      else if(8<=tmp[z][y+i][x+i]&&tmp[z][y+i][x+i]<=14){
         tmp[z][y+i][x+i]=-2;collision[0]++;}
      else collision[0]++;}}
      if(!collision[1]){if(x+i<5 && y-i>=0){
         if(tmp[z][y-i][x+i]==NONE){tmp[z][y-i][x+i]=-1;}
      else if(8<=tmp[z][y-i][x+i]&&tmp[z][y-i][x+i]<=14){
         tmp[z][y-i][x+i]=-2;collision[1]++;}
      else collision[1]++;}}
      if(!collision[2]){if(x-i>=0 && y+i<5){
         if(tmp[z][y+i][x-i]==NONE){tmp[z][y+i][x-i]=-1;}
      else if(8<=tmp[z][y+i][x-i]&&tmp[z][y+i][x-i]<=14){
         tmp[z][y+i][x-i]=-2;collision[2]++;}
      else collision[2]++;}}
      if(!collision[3]){if(x-i>=0 && y-i>=0){
         if(tmp[z][y-i][x-i]==NONE){tmp[z][y-i][x-i]=-1;}
      else if(8<=tmp[z][y-i][x-i]&&tmp[z][y-i][x-i]<=14){
         tmp[z][y-i][x-i]=-2;collision[3]++;}
      else collision[3]++;}}
      //三次元面上方向
      if(!collision[4]){if(z+i<5 && x+i<5){
         if(tmp[z+i][y][x+i]==NONE){tmp[z+i][y][x+i]=-1;}
      else if(8<=tmp[z+i][y][x+i]&&tmp[z+i][y][x+i]<=14){
         tmp[z+i][y][x+i]=-2;collision[4]++;}
      else collision[4]++;}}
      if(!collision[5]){if(z+i<5 && x-i>=0){
         if(tmp[z+i][y][x-i]==NONE){tmp[z+i][y][x-i]=-1;}
      else if(8<=tmp[z+i][y][x-i]&&tmp[z+i][y][x-i]<=14){
         tmp[z+i][y][x-i]=-2;collision[5]++;}
      else collision[5]++;}}
```

```
      if(!collision[6]){if(z+i<5 && y+i<5){
         if(tmp[z+i][y+i][x]==NONE){tmp[z+i][y+i][x]=-1;}
      else if(8<=tmp[z+i][y+i][x]&&tmp[z+i][y+i][x]<=14){
         tmp[z+i][y+i][x]=-2;collision[6]++;}
      else collision[6]++;}}
      if(!collision[7]){if(z+i<5 && y-i>=0){
         if(tmp[z+i][y-i][x]==NONE){tmp[z+i][y-i][x]=-1;}
      else if(8<=tmp[z+i][y-i][x]&&tmp[z+i][y-i][x]<=14){
         tmp[z+i][y-i][x]=-2;collision[7]++;}
      else collision[7]++;}}
      //三次元面下方向
      if(!collision[8]){if(z-i>=0 && x+i<5){
         if(tmp[z-i][y][x+i]==NONE){tmp[z-i][y][x+i]=-1;}
      else if(8<=tmp[z-i][y][x+i]&&tmp[z-i][y][x+i]<=14){
         tmp[z-i][y][x+i]=-2;collision[8]++;}
      else collision[8]++;}}
      if(!collision[9]){if(z-i>=0 && x-i>=0){
         if(tmp[z-i][y][x-i]==NONE){tmp[z-i][y][x-i]=-1;}
      else if(8<=tmp[z-i][y][x-i]&&tmp[z-i][y][x-i]<=14){
         tmp[z-i][y][x-i]=-2;collision[9]++;}
      else collision[9]++;}}
      if(!collision[10]){if(z-i>=0 && y+i<5){
         if(tmp[z-i][y+i][x]==NONE){tmp[z-i][y+i][x]=-1;}
      else if(8<=tmp[z-i][y+i][x]&&tmp[z-i][y+i][x]<=14){
         tmp[z-i][y+i][x]=-2;collision[10]++;}
      else collision[10]++;}}
      if(!collision[11]){if(z-i>=0 && y-i>=0){
         if(tmp[z-i][y-i][x]==NONE){tmp[z-i][y-i][x]=-1;}
      else if(8<=tmp[z-i][y-i][x]&&tmp[z-i][y-i][x]<=14){
         tmp[z-i][y-i][x]=-2;collision[11]++;}
      else collision[11]++;}}
   }
   break;
         }
case W_U:{
   for(int i=1;i<5;i++){
      //三次元面上方向
      if(!collision[0]){if(z+i<5 && x+i<5 && y+i<5){
         if(tmp[z+i][y+i][x+i]==NONE){tmp[z+i][y+i][x+i]=-1;}
      else if(8<=tmp[z+i][y+i][x+i]&&tmp[z+i][y+i][x+i]<=14){
```

```
      tmp[z+i][y+i][x+i]=-2;collision[0]++;}
   else collision[0]++;}}
   if(!collision[1]){if(z+i<5 && x+i<5 && y-i>=0){
      if(tmp[z+i][y-i][x+i]==NONE){tmp[z+i][y-i][x+i]=-1;}
   else if(8<=tmp[z+i][y-i][x+i]&&tmp[z+i][y-i][x+i]<=14){
      tmp[z+i][y-i][x+i]=-2;collision[1]++;}
   else collision[1]++;}}
   if(!collision[2]){if(z+i<5 && x-i>=0 && y+i<5){
      if(tmp[z+i][y+i][x-i]==NONE){tmp[z+i][y+i][x-i]=-1;}
   else if(8<=tmp[z+i][y+i][x-i]&&tmp[z+i][y+i][x-i]<=14){
      tmp[z+i][y+i][x-i]=-2;collision[2]++;}
   else collision[2]++;}}
   if(!collision[3]){if(z+i<5 && x-i>=0 && y-i>=0){
      if(tmp[z+i][y-i][x-i]==NONE){tmp[z+i][y-i][x-i]=-1;}
   else if(8<=tmp[z+i][y-i][x-i]&&tmp[z+i][y-i][x-i]<=14){
      tmp[z+i][y-i][x-i]=-2;collision[3]++;}
   else collision[3]++;}}
   //三次元面下方向
   if(!collision[4]){if(z-i>=0 && x+i<5 && y+i<5){
      if(tmp[z-i][y+i][x+i]==NONE){tmp[z-i][y+i][x+i]=-1;}
   else if(8<=tmp[z-i][y+i][x+i]&&tmp[z-i][y+i][x+i]<=14){
      tmp[z-i][y+i][x+i]=-2;collision[4]++;}
   else collision[4]++;}}
   if(!collision[5]){if(z-i>=0 && x+i<5 && y-i>=0){
      if(tmp[z-i][y-i][x+i]==NONE){tmp[z-i][y-i][x+i]=-1;}
   else if(8<=tmp[z-i][y-i][x+i]&&tmp[z-i][y-i][x+i]<=14){
      tmp[z-i][y-i][x+i]=-2;collision[5]++;}
   else collision[5]++;}}
   if(!collision[6]){if(z-i>=0 && x-i>=0 && y+i<5){
      if(tmp[z-i][y+i][x-i]==NONE){tmp[z-i][y+i][x-i]=-1;}
   else if(8<=tmp[z-i][y+i][x-i]&&tmp[z-i][y+i][x-i]<=14){
      tmp[z-i][y+i][x-i]=-2;collision[6]++;}
   else collision[6]++;}}
   if(!collision[7]){if(z-i>=0 && x-i>=0 && y-i>=0){
      if(tmp[z-i][y-i][x-i]==NONE){tmp[z-i][y-i][x-i]=-1;}
   else if(8<=tmp[z-i][y-i][x-i]&&tmp[z-i][y-i][x-i]<=14){
      tmp[z-i][y-i][x-i]=-2;collision[7]++;}
   else collision[7]++;}}
}
break;
```

```
          }
case W_P:{
   //移動
   if(y+1<5 && tmp[z][y+1][x]==NONE)tmp[z][y+1][x]=-1;
   if(z+1<5 && tmp[z+1][y][x]==NONE)tmp[z+1][y][x]=-1;
   //攻撃
   if(y+1<5 && x-1>=0 && 8<=tmp[z][y+1][x-1] && tmp[z][y+1][x-1]<=14)
      tmp[z][y+1][x-1]=-2;
   if(y+1<5 && x+1<5 && 8<=tmp[z][y+1][x+1] && tmp[z][y+1][x+1]<=14)
      tmp[z][y+1][x+1]=-2;
   if(z+1<5 && x-1>=0 && 8<=tmp[z+1][y][x-1] && tmp[z+1][y][x-1]<=14)
      tmp[z+1][y][x-1]=-2;
   if(z+1<5 && x+1<5 && 8<=tmp[z+1][y][x+1] && tmp[z+1][y][x+1]<=14)
      tmp[z+1][y][x+1]=-2;
   if(z+1<5 && y+1<5 && 8<=tmp[z+1][y+1][x] && tmp[z+1][y+1][x]<=14)
      tmp[z+1][y+1][x]=-2;
   break;
       }
//コマの色が黒（8〜14 の場合）
case B_K:{
   for(int i=0;i<3;i++){
      for(int j=0;j<3;j++){
         for(int k=0;k<3;k++){
            if(tmp[z-1+i][y-1+j][x-1+k]==NONE){
               tmp[z-1+i][y-1+j][x-1+k]=MOV;
            }else if(1<=tmp[z-1+i][y-1+j][x-1+k]&&
               tmp[z-1+i][y-1+j][x-1+k]<=7){
               tmp[z-1+i][y-1+j][x-1+k]=ATT;
            }
         }
      }
   }
   break;
    }
case B_Q:{
   for(int i=1;i<5;i++){
      //ROOK
      //二次元面
      if(!collision[0]){if(x+i<5){
         if(tmp[z][y][x+i]==NONE){tmp[z][y][x+i]=-1;}
```

```
else if(1<=tmp[z][y][x+i]&&tmp[z][y][x+i]<=7){
   tmp[z][y][x+i]=-2;collision[0]++;}
else collision[0]++;}}
if(!collision[1]){if(x-i>=0){
   if(tmp[z][y][x-i]==NONE){tmp[z][y][x-i]=-1;}
else if(1<=tmp[z][y][x-i]&&tmp[z][y][x-i]<=7){
   tmp[z][y][x-i]=-2;collision[1]++;}
else collision[1]++;}}
if(!collision[2]){if(y+i<5){
   if(tmp[z][y+i][x]==NONE){tmp[z][y+i][x]=-1;}
else if(1<=tmp[z][y+i][x]&&tmp[z][y+i][x]<=7){
   tmp[z][y+i][x]=-2;collision[2]++;}
else collision[2]++;}}
if(!collision[3]){if(y-i>=0){
   if(tmp[z][y-i][x]==NONE){tmp[z][y-i][x]=-1;}
else if(1<=tmp[z][y-i][x]&&tmp[z][y-i][x]<=7){
   tmp[z][y-i][x]=-2;collision[3]++;}
else collision[3]++;}}
//三次元面
if(!collision[4]){if(z+i<5){
   if(tmp[z+i][y][x]==NONE){tmp[z+i][y][x]=-1;}
else if(1<=tmp[z+i][y][x]&&tmp[z+i][y][x]<=7){
   tmp[z+i][y][x]=-2;collision[4]++;}
else collision[4]++;}}
if(!collision[5]){if(z-i>=0){
   if(tmp[z-i][y][x]==NONE){tmp[z-i][y][x]=-1;}
else if(1<=tmp[z-i][y][x]&&tmp[z-i][y][x]<=7){
   tmp[z-i][y][x]=-2;collision[5]++;}
else collision[5]++;}}
//BISHOP
//二次元面
if(!collision[6]){if(x+i<5 && y+i<5){
   if(tmp[z][y+i][x+i]==NONE){tmp[z][y+i][x+i]=-1;}
else if(1<=tmp[z][y+i][x+i]&&tmp[z][y+i][x+i]<=7){
   tmp[z][y+i][x+i]=-2;collision[6]++;}
else collision[6]++;}}
if(!collision[7]){if(x+i<5 && y-i>=0){
   if(tmp[z][y-i][x+i]==NONE){tmp[z][y-i][x+i]=-1;}
else if(1<=tmp[z][y-i][x+i]&&tmp[z][y-i][x+i]<=7){
   tmp[z][y-i][x+i]=-2;collision[7]++;}
```

```
else collision[7]++;}}
if(!collision[8]){if(x-i>=0 && y+i<5){
   if(tmp[z][y+i][x-i]==NONE){tmp[z][y+i][x-i]=-1;}
else if(1<=tmp[z][y+i][x-i]&&tmp[z][y+i][x-i]<=7){
   tmp[z][y+i][x-i]=-2;collision[8]++;}
else collision[8]++;}}
if(!collision[9]){if(x-i>=0 && y-i>=0){
   if(tmp[z][y-i][x-i]==NONE){tmp[z][y-i][x-i]=-1;}
else if(1<=tmp[z][y-i][x-i]&&tmp[z][y-i][x-i]<=7){
   tmp[z][y-i][x-i]=-2;collision[9]++;}
else collision[9]++;}}
```

//三次元面上方向

```
if(!collision[10]){if(z+i<5 && x+i<5){
   if(tmp[z+i][y][x+i]==NONE){tmp[z+i][y][x+i]=-1;}
else if(1<=tmp[z+i][y][x+i]&&tmp[z+i][y][x+i]<=7){
   tmp[z+i][y][x+i]=-2;collision[10]++;}
else collision[10]++;}}
if(!collision[11]){if(z+i<5 && x-i>=0){
   if(tmp[z+i][y][x-i]==NONE){tmp[z+i][y][x-i]=-1;}
else if(1<=tmp[z+i][y][x-i]&&tmp[z+i][y][x-i]<=7){
   tmp[z+i][y][x-i]=-2;collision[11]++;}
else collision[11]++;}}
if(!collision[12]){if(z+i<5 && y+i<5){
   if(tmp[z+i][y+i][x]==NONE){tmp[z+i][y+i][x]=-1;}
else if(1<=tmp[z+i][y+i][x]&&tmp[z+i][y+i][x]<=7){
   tmp[z+i][y+i][x]=-2;collision[12]++;}
else collision[12]++;}}
if(!collision[13]){if(z+i<5 && y-i>=0){
   if(tmp[z+i][y-i][x]==NONE){tmp[z+i][y-i][x]=-1;}
else if(1<=tmp[z+i][y-i][x]&&tmp[z+i][y-i][x]<=7){
   tmp[z+i][y-i][x]=-2;collision[13]++;}
else collision[13]++;}}
```

//三次元面下方向

```
if(!collision[14]){if(z-i>=0 && x+i<5){
   if(tmp[z-i][y][x+i]==NONE){tmp[z-i][y][x+i]=-1;}
else if(1<=tmp[z-i][y][x+i]&&tmp[z-i][y][x+i]<=7){
   tmp[z-i][y][x+i]=-2;collision[14]++;}
else collision[14]++;}}
if(!collision[15]){if(z-i>=0 && x-i>=0){
   if(tmp[z-i][y][x-i]==NONE){tmp[z-i][y][x-i]=-1;}
```

```
else if(1<=tmp[z-i][y][x-i]&&tmp[z-i][y][x-i]<=7){
   tmp[z-i][y][x-i]=-2;collision[15]++;}
else collision[15]++;}}
if(!collision[16]){if(z-i>=0 && y+i<5){
   if(tmp[z-i][y+i][x]==NONE){tmp[z-i][y+i][x]=-1;}
else if(1<=tmp[z-i][y+i][x]&&tmp[z-i][y+i][x]<=7){
   tmp[z-i][y+i][x]=-2;collision[16]++;}
else collision[16]++;}}
if(!collision[17]){if(z-i>=0 && y-i>=0){
   if(tmp[z-i][y-i][x]==NONE){tmp[z-i][y-i][x]=-1;}
else if(1<=tmp[z-i][y-i][x]&&tmp[z-i][y-i][x]<=7){
   tmp[z-i][y-i][x]=-2;collision[17]++;}
else collision[17]++;}}
//UNICORN
//三次元面上方向
if(!collision[18]){if(z+i<5 && x+i<5 && y+i<5){
   if(tmp[z+i][y+i][x+i]==NONE){tmp[z+i][y+i][x+i]=-1;}
else if(1<=tmp[z+i][y+i][x+i]&&tmp[z+i][y+i][x+i]<=7){
   tmp[z+i][y+i][x+i]=-2;collision[18]++;}
else collision[18]++;}}
if(!collision[19]){if(z+i<5 && x+i<5 && y-i>=0){
   if(tmp[z+i][y-i][x+i]==NONE){tmp[z+i][y-i][x+i]=-1;}
else if(1<=tmp[z+i][y-i][x+i]&&tmp[z+i][y-i][x+i]<=7){
   tmp[z+i][y-i][x+i]=-2;collision[19]++;}
else collision[19]++;}}
if(!collision[20]){if(z+i<5 && x-i>=0 && y+i<5){
   if(tmp[z+i][y+i][x-i]==NONE){tmp[z+i][y+i][x-i]=-1;}
else if(1<=tmp[z+i][y+i][x-i]&&tmp[z+i][y+i][x-i]<=7){
   tmp[z+i][y+i][x-i]=-2;collision[20]++;}
else collision[20]++;}}
if(!collision[21]){if(z+i<5 && x-i>=0 && y-i>=0){
   if(tmp[z+i][y-i][x-i]==NONE){tmp[z+i][y-i][x-i]=-1;}
else if(1<=tmp[z+i][y-i][x-i]&&tmp[z+i][y-i][x-i]<=7){
   tmp[z+i][y-i][x-i]=-2;collision[21]++;}
else collision[21]++;}}
//三次元面下方向
if(!collision[22]){if(z-i>=0 && x+i<5 && y+i<5){
   if(tmp[z-i][y+i][x+i]==NONE){tmp[z-i][y+i][x+i]=-1;}
else if(1<=tmp[z-i][y+i][x+i]&&tmp[z-i][y+i][x+i]<=7){
   tmp[z-i][y+i][x+i]=-2;collision[22]++;}
```

```
      else collision[22]++;}}
      if(!collision[23]){if(z-i>=0 && x+i<5 && y-i>=0){
         if(tmp[z-i][y-i][x+i]==NONE){tmp[z-i][y-i][x+i]=-1;}
      else if(1<=tmp[z-i][y-i][x+i]&&tmp[z-i][y-i][x+i]<=7){
         tmp[z-i][y-i][x+i]=-2;collision[23]++;}
      else collision[23]++;}}
      if(!collision[24]){if(z-i>=0 && x-i>=0 && y+i<5){
         if(tmp[z-i][y+i][x-i]==NONE){tmp[z-i][y+i][x-i]=-1;}
      else if(1<=tmp[z-i][y+i][x-i]&&tmp[z-i][y+i][x-i]<=7){
         tmp[z-i][y+i][x-i]=-2;collision[24]++;}
      else collision[24]++;}}
      if(!collision[25]){if(z-i>=0 && x-i>=0 && y-i>=0){
         if(tmp[z-i][y-i][x-i]==NONE){tmp[z-i][y-i][x-i]=-1;}
      else if(1<=tmp[z-i][y-i][x-i]&&tmp[z-i][y-i][x-i]<=7){
         tmp[z-i][y-i][x-i]=-2;collision[25]++;}
      else collision[25]++;}}
   }
   break;
        }
case B_N:{
   if(y+1<5 && x+2<5){if(tmp[z][y+1][x+2]==NONE)
      tmp[z][y+1][x+2]=-1;
   else if(1<=tmp[z][y+1][x+2] && tmp[z][y+1][x+2]<=7)
      tmp[z][y+1][x+2]=-2;}
   if(y+1<5 && x-2>=0){if(tmp[z][y+1][x-2]==NONE)
      tmp[z][y+1][x-2]=-1;
   else if(1<=tmp[z][y+1][x-2] && tmp[z][y+1][x-2]<=7)
      tmp[z][y+1][x-2]=-2;}
   if(y-1>=0 && x+2<5){if(tmp[z][y-1][x+2]==NONE)
      tmp[z][y-1][x+2]=-1;
   else if(1<=tmp[z][y-1][x+2] && tmp[z][y-1][x+2]<=7)
      tmp[z][y-1][x+2]=-2;}
   if(y-1>=0 && x-2>=0){if(tmp[z][y-1][x-2]==NONE)
      tmp[z][y-1][x-2]=-1;
   else if(1<=tmp[z][y-1][x-2] && tmp[z][y-1][x-2]<=7)
      tmp[z][y-1][x-2]=-2;}
   if(y+2<5 && x+1<5){if(tmp[z][y+2][x+1]==NONE)
      tmp[z][y+2][x+1]=-1;
   else if(1<=tmp[z][y+2][x+1] && tmp[z][y+2][x+1]<=7)
      tmp[z][y+2][x+1]=-2;}
```

```
if(y+2<5 && x-1>=0){if(tmp[z][y+2][x-1]==NONE)
   tmp[z][y+2][x-1]=-1;
else if(1<=tmp[z][y+2][x-1] && tmp[z][y+2][x-1]<=7)
   tmp[z][y+2][x-1]=-2;}
if(y-2>=0 && x+1<5){if(tmp[z][y-2][x+1]==NONE)
   tmp[z][y-2][x+1]=-1;
else if(1<=tmp[z][y-2][x+1] && tmp[z][y-2][x+1]<=7)
   tmp[z][y-2][x+1]=-2;}
if(y-2>=0 && x-1>=0){if(tmp[z][y-2][x-1]==NONE)
   tmp[z][y-2][x-1]=-1;
else if(1<=tmp[z][y-2][x-1] && tmp[z][y-2][x-1]<=7)
   tmp[z][y-2][x-1]=-2;}

if(z+1<5 && x+2<5){if(tmp[z+1][y][x+2]==NONE)
   tmp[z+1][y][x+2]=-1;
else if(1<=tmp[z+1][y][x+2] && tmp[z+1][y][x+2]<=7)
   tmp[z+1][y][x+2]=-2;}
if(z+1<5 && x-2>=0){if(tmp[z+1][y][x-2]==NONE)
   tmp[z+1][y][x-2]=-1;
else if(1<=tmp[z+1][y][x-2] && tmp[z+1][y][x-2]<=7)
   tmp[z+1][y][x-2]=-2;}
if(z-1>=0 && x+2<5){if(tmp[z-1][y][x+2]==NONE)
   tmp[z-1][y][x+2]=-1;
else if(1<=tmp[z-1][y][x+2] && tmp[z-1][y][x+2]<=7)
   tmp[z-1][y][x+2]=-2;}
if(z-1>=0 && x-2>=0){if(tmp[z-1][y][x-2]==NONE)
   tmp[z-1][y][x-2]=-1;
else if(1<=tmp[z-1][y][x-2] && tmp[z-1][y][x-2]<=7)
   tmp[z-1][y][x-2]=-2;}
if(z+1<5 && y+2<5){if(tmp[z+1][y+2][x]==NONE)
   tmp[z+1][y+2][x]=-1;
else if(1<=tmp[z+1][y+2][x] && tmp[z+1][y+2][x]<=7)
   tmp[z+1][y+2][x]=-2;}
if(z+1<5 && y-2>=0){if(tmp[z+1][y-2][x]==NONE)
   tmp[z+1][y-2][x]=-1;
else if(1<=tmp[z+1][y-2][x] && tmp[z+1][y-2][x]<=7)
   tmp[z+1][y-2][x]=-2;}
if(z-1>=0 && y+2<5){if(tmp[z-1][y+2][x]==NONE)
   tmp[z-1][y+2][x]=-1;
else if(1<=tmp[z-1][y+2][x] && tmp[z-1][y+2][x]<=7)
```

```
   tmp[z-1][y+2][x]=-2;}
if(z-1>=0 && y-2>=0){if(tmp[z-1][y-2][x]==NONE)
   tmp[z-1][y-2][x]=-1;
else if(1<=tmp[z-1][y-2][x] && tmp[z-1][y-2][x]<=7)
   tmp[z-1][y-2][x]=-2;}

if(z+2<5 && x+1<5){if(tmp[z+2][y][x+1]==NONE)
   tmp[z+2][y][x+1]=-1;
else if(1<=tmp[z+2][y][x+1] && tmp[z+2][y][x+1]<=7)
   tmp[z+2][y][x+1]=-2;}
if(z+2<5 && x-1>=0){if(tmp[z+2][y][x-1]==NONE)
   tmp[z+2][y][x-1]=-1;
else if(1<=tmp[z+2][y][x-1] && tmp[z+2][y][x-1]<=7)
   tmp[z+2][y][x-1]=-2;}
if(z-2>=0 && x+1<5){if(tmp[z-2][y][x+1]==NONE)
   tmp[z-2][y][x+1]=-1;
else if(1<=tmp[z-2][y][x+1] && tmp[z-2][y][x+1]<=7)
   tmp[z-2][y][x+1]=-2;}
if(z-2>=0 && x-1>=0){if(tmp[z-2][y][x-1]==NONE)
   tmp[z-2][y][x-1]=-1;
else if(1<=tmp[z-2][y][x-1] && tmp[z-2][y][x-1]<=7)
   tmp[z-2][y][x-1]=-2;}
if(z+2<5 && y+1<5){if(tmp[z+2][y+1][x]==NONE)
   tmp[z+2][y+1][x]=-1;
else if(1<=tmp[z+2][y+1][x] && tmp[z+2][y+1][x]<=7)
   tmp[z+2][y+1][x]=-2;}
if(z+2<5 && y-1>=0){if(tmp[z+2][y-1][x]==NONE)
   tmp[z+2][y-1][x]=-1;
else if(1<=tmp[z+2][y-1][x] && tmp[z+2][y-1][x]<=7)
   tmp[z+2][y-1][x]=-2;}
if(z-2>=0 && y+1<5){if(tmp[z-2][y+1][x]==NONE)
   tmp[z-2][y+1][x]=-1;
else if(1<=tmp[z-2][y+1][x] && tmp[z-2][y+1][x]<=7)
   tmp[z-2][y+1][x]=-2;}
if(z-2>=0 && y-1>=0){if(tmp[z-2][y-1][x]==NONE)
   tmp[z-2][y-1][x]=-1;
else if(1<=tmp[z-2][y-1][x] && tmp[z-2][y-1][x]<=7)
   tmp[z-2][y-1][x]=-2;}
break;
      }
```

```
case B_R:{
   for(int i=1;i<5;i++){
      //二次元面
      if(!collision[0]){if(x+i<5){
         if(tmp[z][y][x+i]==NONE){tmp[z][y][x+i]=-1;}
      else if(1<=tmp[z][y][x+i]&&tmp[z][y][x+i]<=7){
         tmp[z][y][x+i]=-2;collision[0]++;}
      else collision[0]++;}}
      if(!collision[1]){if(x-i>=0){
         if(tmp[z][y][x-i]==NONE){tmp[z][y][x-i]=-1;}
      else if(1<=tmp[z][y][x-i]&&tmp[z][y][x-i]<=7){
         tmp[z][y][x-i]=-2;collision[1]++;}
      else collision[1]++;}}
      if(!collision[2]){if(y+i<5){
         if(tmp[z][y+i][x]==NONE){tmp[z][y+i][x]=-1;}
      else if(1<=tmp[z][y+i][x]&&tmp[z][y+i][x]<=7){
         tmp[z][y+i][x]=-2;collision[2]++;}
      else collision[2]++;}}
      if(!collision[3]){if(y-i>=0){
         if(tmp[z][y-i][x]==NONE){tmp[z][y-i][x]=-1;}
      else if(1<=tmp[z][y-i][x]&&tmp[z][y-i][x]<=7){
         tmp[z][y-i][x]=-2;collision[3]++;}
      else collision[3]++;}}
      //三次元面
      if(!collision[4]){if(z+i<5){
         if(tmp[z+i][y][x]==NONE){tmp[z+i][y][x]=-1;}
      else if(1<=tmp[z+i][y][x]&&tmp[z+i][y][x]<=7){
         tmp[z+i][y][x]=-2;collision[4]++;}
      else collision[4]++;}}
      if(!collision[5]){if(z-i>=0){
         if(tmp[z-i][y][x]==NONE){tmp[z-i][y][x]=-1;}
      else if(1<=tmp[z-i][y][x]&&tmp[z-i][y][x]<=7){
         tmp[z-i][y][x]=-2;collision[5]++;}
      else collision[5]++;}}
   }
   break;
       }
case B_B:{
   for(int i=1;i<5;i++){
      //二次元面
```

```
if(!collision[0]){if(x+i<5 && y+i<5){
   if(tmp[z][y+i][x+i]==NONE){tmp[z][y+i][x+i]=-1;}
else if(1<=tmp[z][y+i][x+i]&&tmp[z][y+i][x+i]<=7){
   tmp[z][y+i][x+i]=-2;collision[0]++;}
else collision[0]++;}}
if(!collision[1]){if(x+i<5 && y-i>=0){
   if(tmp[z][y-i][x+i]==NONE){tmp[z][y-i][x+i]=-1;}
else if(1<=tmp[z][y-i][x+i]&&tmp[z][y-i][x+i]<=7){
   tmp[z][y-i][x+i]=-2;collision[1]++;}
else collision[1]++;}}
if(!collision[2]){if(x-i>=0 && y+i<5){
   if(tmp[z][y+i][x-i]==NONE){tmp[z][y+i][x-i]=-1;}
else if(1<=tmp[z][y+i][x-i]&&tmp[z][y+i][x-i]<=7){
   tmp[z][y+i][x-i]=-2;collision[2]++;}
else collision[2]++;}}
if(!collision[3]){if(x-i>=0 && y-i>=0){
   if(tmp[z][y-i][x-i]==NONE){tmp[z][y-i][x-i]=-1;}
else if(1<=tmp[z][y-i][x-i]&&tmp[z][y-i][x-i]<=7){
   tmp[z][y-i][x-i]=-2;collision[3]++;}
else collision[3]++;}}
//三次元面上方向
if(!collision[4]){if(z+i<5 && x+i<5){
   if(tmp[z+i][y][x+i]==NONE){tmp[z+i][y][x+i]=-1;}
else if(1<=tmp[z+i][y][x+i]&&tmp[z+i][y][x+i]<=7){
   tmp[z+i][y][x+i]=-2;collision[4]++;}
else collision[4]++;}}
if(!collision[5]){if(z+i<5 && x-i>=0){
   if(tmp[z+i][y][x-i]==NONE){tmp[z+i][y][x-i]=-1;}
else if(1<=tmp[z+i][y][x-i]&&tmp[z+i][y][x-i]<=7){
   tmp[z+i][y][x-i]=-2;collision[5]++;}
else collision[5]++;}}
if(!collision[6]){if(z+i<5 && y+i<5){
   if(tmp[z+i][y+i][x]==NONE){tmp[z+i][y+i][x]=-1;}
else if(1<=tmp[z+i][y+i][x]&&tmp[z+i][y+i][x]<=7){
   tmp[z+i][y+i][x]=-2;collision[6]++;}
else collision[6]++;}}
if(!collision[7]){if(z+i<5 && y-i>=0){
   if(tmp[z+i][y-i][x]==NONE){tmp[z+i][y-i][x]=-1;}
else if(1<=tmp[z+i][y-i][x]&&tmp[z+i][y-i][x]<=7){
   tmp[z+i][y-i][x]=-2;collision[7]++;}
```

```
      else collision[7]++;}}
      //三次元面下方向
      if(!collision[8]){if(z-i>=0 && x+i<5){
         if(tmp[z-i][y][x+i]==NONE){tmp[z-i][y][x+i]=-1;}
      else if(1<=tmp[z-i][y][x+i]&&tmp[z-i][y][x+i]<=7){
         tmp[z-i][y][x+i]=-2;collision[8]++;}
      else collision[8]++;}}
      if(!collision[9]){if(z-i>=0 && x-i>=0){
         if(tmp[z-i][y][x-i]==NONE){tmp[z-i][y][x-i]=-1;}
      else if(1<=tmp[z-i][y][x-i]&&tmp[z-i][y][x-i]<=7){
         tmp[z-i][y][x-i]=-2;collision[9]++;}
      else collision[9]++;}}
      if(!collision[10]){if(z-i>=0 && y+i<5){
         if(tmp[z-i][y+i][x]==NONE){tmp[z-i][y+i][x]=-1;}
      else if(1<=tmp[z-i][y+i][x]&&tmp[z-i][y+i][x]<=7){
         tmp[z-i][y+i][x]=-2;collision[10]++;}
      else collision[10]++;}}
      if(!collision[11]){if(z-i>=0 && y-i>=0){
         if(tmp[z-i][y-i][x]==NONE){tmp[z-i][y-i][x]=-1;}
      else if(1<=tmp[z-i][y-i][x]&&tmp[z-i][y-i][x]<=7){
         tmp[z-i][y-i][x]=-2;collision[11]++;}
      else collision[11]++;}}
   }
   break;
         }
case B_U:{
   for(int i=1;i<5;i++){
      //三次元面上方向
      if(!collision[0]){if(z+i<5 && x+i<5 && y+i<5){
         if(tmp[z+i][y+i][x+i]==NONE){tmp[z+i][y+i][x+i]=-1;}
      else if(1<=tmp[z+i][y+i][x+i]&&tmp[z+i][y+i][x+i]<=7){
         tmp[z+i][y+i][x+i]=-2;collision[0]++;}
      else collision[0]++;}}
      if(!collision[1]){if(z+i<5 && x+i<5 && y-i>=0){
         if(tmp[z+i][y-i][x+i]==NONE){tmp[z+i][y-i][x+i]=-1;}
      else if(1<=tmp[z+i][y-i][x+i]&&tmp[z+i][y-i][x+i]<=7){
         tmp[z+i][y-i][x+i]=-2;collision[1]++;}
      else collision[1]++;}}
      if(!collision[2]){if(z+i<5 && x-i>=0 && y+i<5){
         if(tmp[z+i][y+i][x-i]==NONE){tmp[z+i][y+i][x-i]=-1;}
```

```
        else if(1<=tmp[z+i][y+i][x-i]&&tmp[z+i][y+i][x-i]<=7){
           tmp[z+i][y+i][x-i]=-2;collision[2]++;}
        else collision[2]++;}}
        if(!collision[3]){if(z+i<5 && x-i>=0 && y-i>=0){
           if(tmp[z+i][y-i][x-i]==NONE){tmp[z+i][y-i][x-i]=-1;}
        else if(1<=tmp[z+i][y-i][x-i]&&tmp[z+i][y-i][x-i]<=7){
           tmp[z+i][y-i][x-i]=-2;collision[3]++;}
        else collision[3]++;}}
        //三次元面下方向
        if(!collision[4]){if(z-i>=0 && x+i<5 && y+i<5){
           if(tmp[z-i][y+i][x+i]==NONE){tmp[z-i][y+i][x+i]=-1;}
        else if(1<=tmp[z-i][y+i][x+i]&&tmp[z-i][y+i][x+i]<=7){
           tmp[z-i][y+i][x+i]=-2;collision[4]++;}
        else collision[4]++;}}
        if(!collision[5]){if(z-i>=0 && x+i<5 && y-i>=0){
           if(tmp[z-i][y-i][x+i]==NONE){tmp[z-i][y-i][x+i]=-1;}
        else if(1<=tmp[z-i][y-i][x+i]&&tmp[z-i][y-i][x+i]<=7){
           tmp[z-i][y-i][x+i]=-2;collision[5]++;}
        else collision[5]++;}}
        if(!collision[6]){if(z-i>=0 && x-i>=0 && y+i<5){
           if(tmp[z-i][y+i][x-i]==NONE){tmp[z-i][y+i][x-i]=-1;}
        else if(1<=tmp[z-i][y+i][x-i]&&tmp[z-i][y+i][x-i]<=7){
           tmp[z-i][y+i][x-i]=-2;collision[6]++;}
        else collision[6]++;}}
        if(!collision[7]){if(z-i>=0 && x-i>=0 && y-i>=0){
           if(tmp[z-i][y-i][x-i]==NONE){tmp[z-i][y-i][x-i]=-1;}
        else if(1<=tmp[z-i][y-i][x-i]&&tmp[z-i][y-i][x-i]<=7){
           tmp[z-i][y-i][x-i]=-2;collision[7]++;}
        else collision[7]++;}}
     }
     break;
           }
case B_P:{
     //移動
     if(y-1>=0 && tmp[z][y-1][x]==NONE)tmp[z][y-1][x]=-1;
     if(z-1>=0 && tmp[z-1][y][x]==NONE)tmp[z-1][y][x]=-1;
     //攻撃
     if(y-1>=0 && x-1>=0 && 1<=tmp[z][y-1][x-1] && tmp[z][y-1][x-1]<=7)
        tmp[z][y-1][x-1]=-2;
     if(y-1>=0 && x+1<5 && 1<=tmp[z][y-1][x+1] && tmp[z][y-1][x+1]<=7)
```

```
            tmp[z][y-1][x+1]=-2;
         if(z-1>=0 && x-1>=0 && 1<=tmp[z-1][y][x-1] && tmp[z-1][y][x-1]<=7)
            tmp[z-1][y][x-1]=-2;
         if(z-1>=0 && x+1<5 && 1<=tmp[z-1][y][x+1] && tmp[z-1][y][x+1]<=7)
            tmp[z-1][y][x+1]=-2;
         if(z-1>=0 && y-1>=0 && 1<=tmp[z-1][y-1][x] && tmp[z-1][y-1][x]<=7)
            tmp[z-1][y-1][x]=-2;
         break;
             }
   }
}
/**
 * 指定色のキングに対してチェックが起きているかを確認する関数
 * @param color:1〜7 は白、8〜14 は黒
 * @return bool:チェックされるならば true
 */
bool checkCheck(int color){
   int i,j,k;
   for(i=0;i<5;i++){
      for(j=0;j<5;j++){
         for(k=0;k<5;k++){
            if(board[k][j][i]==color){
               if(1<=color&&color<=7){
                  for(int a=0;a<5;a++){
                     for(int b=0;b<5;b++){
                        for(int c=0;c<5;c++){
                           if(8<=board[c][b][a]&&board[c][b][a]<=14){
                              moveCheck(a,b,c);
                              if(tmp[k][j][i]==-2)return true;
                           }
                        }
                     }
                  }
               }else{
                  for(int a=0;a<5;a++){
                     for(int b=0;b<5;b++){
                        for(int c=0;c<5;c++){
                           if(1<=board[c][b][a]&&board[c][b][a]<=7){
                              moveCheck(a,b,c);
                              if(tmp[k][j][i]==-2)return true;
```

```
                        }
                    }
                }
            }
        }
    }
}
    return false;
}
/**
 * 指定色のキングに対してチェックメイトが起きているかを確認する関数
 * @param color:1〜7 は白、8〜14 は黒
 * @return bool:チェックされるならば true
 */
bool checkmateCheck(int turn){
    if(currentMode!=1)return false;
    int t1,t2;
    bool check=true;
    if(turn==1){
        for(int i=0;i<5;i++){
            for(int j=0;j<5;j++){
                for(int k=0;k<5;k++){
                    if(7<board[k][j][i]&&board[k][j][i]<15){
                        t1=board[k][j][i];
                        moveCheck(i,j,k);
                        for(int i2=0;i2<5;i2++){
                            for(int j2=0;j2<5;j2++){
                                for(int k2=0;k2<5;k2++){
                                    if(tmp[k2][j2][i2]<0){
                                        t2=board[k2][j2][i2];
                                        board[k2][j2][i2]=t1;
                                        board[k][j][i]=0;
                                        if(!checkCheck(8))check=false;
                                        board[k2][j2][i2]=t2;
                                        board[k][j][i]=t1;
                                        moveCheck(i,j,k);
                                    }
                                }
```

```
                    }
                }
            }
        }
    }
  }else{
    for(int i=0;i<5;i++){
      for(int j=0;j<5;j++){
        for(int k=0;k<5;k++){
          if(0<board[k][j][i]&&board[k][j][i]<8){
            t1=board[k][j][i];
            moveCheck(i,j,k);
            for(int i2=0;i2<5;i2++){
              for(int j2=0;j2<5;j2++){
                for(int k2=0;k2<5;k2++){
                  if(tmp[k2][j2][i2]<0){
                    t2=board[k2][j2][i2];
                    board[k2][j2][i2]=t1;
                    board[k][j][i]=0;
                    if(!checkCheck(1))check=false;
                    board[k2][j2][i2]=t2;
                    board[k][j][i]=t1;
                    moveCheck(i,j,k);
                  }
                }
              }
            }
          }
        }
      }
    }
  }
  if(check)checkmateCall=true;
  winner = turn;
  return check;
}
/**
 * 勝敗がついたかを確認する関数（勝者を winner に格納）
 * @param void
```

```
 * @return bool:キングが片方だけしかないなら true
 */
bool winCheck(){
   if(checkmateCheck(currentTurn))return true;
   int c=0;
   for(int i=0;i<5;i++){
      for(int j=0;j<5;j++){
         for(int k=0;k<5;k++){
            if(board[i][j][k]==W_K)c+=W_K;
            if(board[i][j][k]==B_K)c+=B_K;
         }
      }
   }
   if(c==W_K){
      winner=1;
      return true;
   }else if(c==B_K){
      winner=2;
      return true;
   }else{
      return false;
   }
}
/**
 * ステールメイトが起きているかを確認する関数
 * @param color:1〜7 は白、8〜14 は黒
 * @return bool:ステールメイトならなら true
 */
bool stalemateCheck(int color){
   if(checkCheck(color))return false;
   for(int i=0;i<5;i++){
      for(int j=0;j<5;j++){
         for(int k=0;k<5;k++){
            if(1<=color&&color<=7&&1<=board[k][j][i]&&board[k][j][i]<=7){
               moveCheck(i,j,k);
               for(int a=0;a<5;a++){
                  for(int b=0;b<5;b++){
                     for(int c=0;c<5;c++){
                        if(tmp[a][b][c]==-1||tmp[a][b][c]==-2)return false;
                     }
```

```
                }
            }
        }
        if(8<=color&&color<=14&&8<=board[k][j][i]&&board[k][j][i]<=14){
            moveCheck(i,j,k);
            for(int a=0;a<5;a++){
                for(int b=0;b<5;b++){
                    for(int c=0;c<5;c++){
                        if(tmp[a][b][c]==-1||tmp[a][b][c]==-2)return false;
                    }
                }
            }
        }
    }
  }
  return true;
}
/**
 * プロモーションが起きるかどうかを確認する関数
 * @param color:1〜7 は白、8〜14 は黒
 * @return bool:プロモーションが起きるなら true
 */
bool promotionCheck(int color){
  if(1<=color&&color<=7){
    for(int i=0;i<5;i++){
      if(board[4][4][i]==color)return true;
    }
  }else{
    for(int i=0;i<5;i++){
      if(board[0][0][i]==color)return true;
    }
  }
  return false;
}
```

B.4 main.cpp

```
#include "my3dlib.h"
```

```
//グローバル変数ここから
//前回のループの終了時間
DWORD lasttime;
//ループ内の経過時間
float looptime = 0;
//モデル管理配列識別子
int modelindex[20],bar,MOV,ATT,selectBox;
//カメラの移動
float mx = 0.0f, my = 0.0f, mz = 0.0f;
//セレクトボックスの表示位置
float sx=0.0f,sy=0.0f,sz=0.0f;
//セレクト座標移動前
int sx1=0,sy1=0,sz1=0;
//セレクト座標移動先
int sx2=0,sy2=0,sz2=0;
//Enter キーと Backspace キーの状態
bool selectButton=false,cancelButton=false;
float selectButton1=0.0f,cancelButton1=0.0f;
//カメラの回転速度
float speed = 5.0f;
//フォントハンドル
int hgamefont;
//チェックの状態
bool checkCall=false;
//ゲームの再スタートスイッチ
bool restartCall=false;
//チェックメイトの状態
bool checkmateCall=false;
//ゲームの状態
int currentMode=1;
/*
1-コマ選択
2-移動先選択
3-昇格処理
4-終了・リセット
*/

enum{
QUIT,   //ゲーム終了
```

```
T_W,   //白のターン
T_B   //黒のターン
};
//現在のターン，先手は白
int currentTurn=T_W;
//勝者１が白，２が黒，０で引き分け
int winner=0;
//グローバル変数ここまで

/**
 * ゲームの処理・モデルの描画を行う関数
 * @param void
 * @return void
 */
void GameMain(){
   const char *keys = GetKeyState();

   if( keys != NULL){
      //Shift キーを押しながらでカメラ移動
      if(keys[DIK_LSHIFT]&0x80 || keys[DIK_RSHIFT]&0x80){
         if( keys[DIK_UP]&0x80 )
            my = my + speed * looptime;
         if( keys[DIK_DOWN]&0x80 )
            my = my - speed * looptime;
         if( keys[DIK_LEFT]&0x80 )
            mz = mz - speed * looptime;
         if( keys[DIK_RIGHT]&0x80 )
            mz = mz + speed * looptime;
         if( keys[DIK_ESCAPE]&0x80 || keys[DIK_BACK]&0x80 )
            mx=my=mz=0.0f;
         if( keys[DIK_SLASH]&0x80 )
            mx = mx + speed * looptime;
         if( keys[DIK_BACKSLASH]&0x80 )
            mx = mx - speed * looptime;
         if(mx>0.0f)mx=0.0f;else if(mx<-7.0f)mx=-7.0f;
         if(my>4.0f)my=4.0f;else if(my<-8.0f)my=-8.0f;
         if(mz>6.0f)mz=6.0f;else if(mz<-6.0f)mz=-6.0f;
      }else if(currentMode!=3&&currentMode!=4){
         //セレクトボックスの移動
         if( keys[DIK_UP]&0x80 )
```

```
            sx = sx - 2.0f * speed * looptime;
        if( keys[DIK_DOWN]&0x80 )
            sx = sx + 2.0f * speed * looptime;
        if( keys[DIK_LEFT]&0x80 )
            sz = sz - 2.0f * speed * looptime;
        if( keys[DIK_RIGHT]&0x80 )
            sz = sz + 2.0f * speed * looptime;
        if( keys[DIK_SLASH]&0x80 )
            sy = sy + 2.0f * speed * looptime;
        if( keys[DIK_BACKSLASH]&0x80 )
            sy = sy - 2.0f * speed * looptime;
        if( keys[DIK_RETURN]&0x80 )
            selectButton1 = selectButton1 + speed * looptime;
        if( keys[DIK_ESCAPE]&0x80||keys[DIK_BACK]&0x80 )
            cancelButton1 = cancelButton1 + speed * looptime;
        if(sx>2.0f)sx=2.0f;else if(sx<-2.0f)sx=-2.0f;
        if(sy>2.0f)sy=2.0f;else if(sy<-2.0f)sy=-2.0f;
        if(sz>2.0f)sz=2.0f;else if(sz<-2.0f)sz=-2.0f;
    }
}
if( !(keys[DIK_RETURN]&0x80) &&
    selectButton1>0.0f){selectButton1 = 0.0f;selectButton=true;}
if( !(keys[DIK_ESCAPE]&0x80||keys[DIK_BACK]&0x80) &&
    cancelButton1>0.0f){cancelButton1 = 0.0f;cancelButton=true;}

//視点変更
D3DXVECTOR3 vEyePt( 12.0f+mx, 4.0f+my,2.0f+mz );
D3DXVECTOR3 vLookatPt( 2.0f, 2.0f, 2.0f );
D3DXVECTOR3 vUpVec( 0.0f, 1.0f, 0.0f );
D3DXMATRIXA16 matView;
D3DXMatrixLookAtLH( &matView, &vEyePt, &vLookatPt, &vUpVec );
g_pd3dDevice->SetTransform( D3DTS_VIEW, &matView );

if(winCheck()){   //勝敗確認
    currentMode=4;
    g_ptextsprite->Begin(D3DXSPRITE_ALPHABLEND|D3DXSPRITE_SORT_TEXTURE);
    RECT rc={0,400,640,520};
    if(winner==1){
        g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,_T("白の勝利です."),
            -1,&rc,DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
```

```
   }else if(winner==2){
      g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,_T("黒の勝利です."),
         -1,&rc,DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
   }
   g_ptextsprite->End();
   restartCall=true;
}

//フィールド表示
for(int i=0;i<6;i++){
   for(int j=0;j<6;j++){
      D3DXMATRIXA16 matWorld1,matWorld2;
      D3DXMatrixTranslation( &matWorld1,2.0f,(float)j,(float)-i+4.5f);
      g_pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld1 );
      RenderModel(bar);
      D3DXMatrixTranslation( &matWorld1,(float)j-0.5f,(float)i,2.0f);
      D3DXMatrixRotationY(&matWorld2,D3DX_PI/2);
      matWorld2 *= matWorld1;
      g_pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld2 );
      RenderModel(bar);
      D3DXMatrixTranslation( &matWorld1,(float)j-0.5f,2.5f,(float)-i+4.5f);
      D3DXMatrixRotationZ(&matWorld2,D3DX_PI/2);
      matWorld2 *= matWorld1;
      g_pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld2 );
      RenderModel(bar);
   }
}

//駒の選択中ならばステールメイトをチェックする
if(currentMode==1){
   if((currentTurn==T_W&&stalemateCheck(T_W))||
      (currentTurn==T_B&&stalemateCheck(T_B))){
      g_ptextsprite->Begin(D3DXSPRITE_ALPHABLEND|D3DXSPRITE_SORT_TEXTURE);
      RECT rc={0,400,640,520};
      currentMode=4;
      g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,_T("ステールメイトです."),
         -1,&rc,DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
      restartCall=true;
      g_ptextsprite->End();
   }
```

```
}
//駒の選択中に Enter キーが押された
if(selectButton&&currentMode==1){
   sx1=(int)sx+2;
   sy1=(int)sy+2;
   sz1=(int)sz+2;
   if((currentTurn==T_W&&0<board[sy1][sz1][sx1]&&board[sy1][sz1][sx1]<8)||
      (currentTurn==T_B&&7<board[sy1][sz1][sx1]&&board[sy1][sz1][sx1]<15))
      currentMode=2;
   moveCheck(sx1,sz1,sy1);
}else if(selectButton&&currentMode==2){
   //移動先の選択中に Enter キーが押された
   //移動後チェックされるならば戻る
   //移動後昇格するならばゲーム状態を遷移
   sx2=(int)sx+2;
   sy2=(int)sy+2;
   sz2=(int)sz+2;
   currentMode=1;
   if(tmp[sy2][sz2][sx2]<0){
      int t=board[sy2][sz2][sx2];
      board[sy2][sz2][sx2]=board[sy1][sz1][sx1];
      board[sy1][sz1][sx1]=0;
      if(currentTurn==T_W){
         if(checkCheck(1)){
            setTimer(0,3000);
            checkCall=true;
            board[sy1][sz1][sx1]=board[sy2][sz2][sx2];
            board[sy2][sz2][sx2]=t;
            if(currentTurn==T_W)currentTurn=T_B;
            else if(currentTurn==T_B)currentTurn=T_W;
         }
         if(promotionCheck(7)){
            currentMode=3;
         }
      }else{
         if(checkCheck(8)){
            setTimer(0,3000);
            checkCall=true;
            board[sy1][sz1][sx1]=board[sy2][sz2][sx2];
            board[sy2][sz2][sx2]=t;
```

```
            if(currentTurn==T_W)currentTurn=T_B;
            else if(currentTurn==T_B)currentTurn=T_W;
         }
         if(promotionCheck(14)){
            currentMode=3;
         }
      }
      if(currentTurn==T_W)currentTurn=T_B;
      else if(currentTurn==T_B)currentTurn=T_W;
   }
}
if(cancelButton==1&&currentMode==2){
   currentMode=1;
}
//ボードの表示
if(currentMode==1||currentMode==3||currentMode==4){   //移動するコマを選択
   for(int i=0;i<5;i++){
      for(int j=0;j<5;j++){
         for(int k=0;k<5;k++){
            if(board[k][j][i]!=0){
               D3DXMATRIXA16 matWorld1,matWorld2;
               D3DXMatrixTranslation( &matWorld1,(float)i,(float)k,(float)j);
               D3DXMatrixScaling( &matWorld2,1.0f/4.0f,1.0f/4.0f,1.0f/4.0f);
               matWorld2 *= matWorld1;
               g_pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld2 );
               RenderModel(modelindex[board[k][j][i]]);
            }
         }
      }
   }
}else if(currentMode==2){   //移動先を選択
   for(int i=0;i<5;i++){
      for(int j=0;j<5;j++){
         for(int k=0;k<5;k++){
            if(tmp[k][j][i]!=0){
               D3DXMATRIXA16 matWorld1,matWorld2;
               if(0<board[k][j][i]&&board[k][j][i]<15){
                  D3DXMatrixTranslation( &matWorld1,(float)i,(float)k,(float)j);
                  D3DXMatrixScaling( &matWorld2,1.0f/4.0f,1.0f/4.0f,1.0f/4.0f);
                  matWorld2 *= matWorld1;
```

```
                g_pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld2 );
                RenderModel(modelindex[board[k][j][i]]);
            }
            D3DXMatrixTranslation( &matWorld1,(float)i,(float)k+0.5f,(float)j);
            D3DXMatrixScaling( &matWorld2,1.0f,1.0f,1.0f);
            matWorld2 *= matWorld1;
            g_pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld2 );
            //アルファブレンディングの有効化
            g_pd3dDevice->SetRenderState(D3DRS_ALPHABLENDENABLE,TRUE);
            g_pd3dDevice->SetRenderState(D3DRS_SRCBLEND,D3DBLEND_BLENDFACTOR);
            g_pd3dDevice->SetRenderState(D3DRS_DESTBLEND,D3DBLEND_ONE);

            if(tmp[k][j][i]==-1){
                RenderModel(MOV);
            }else if(tmp[k][j][i]==-2){
                RenderModel(ATT);
            }
            g_pd3dDevice->SetRenderState(D3DRS_ALPHABLENDENABLE,FALSE);
        }
      }
    }
  }
}

//セレクトボックスの表示

//アルファブレンディングの有効化
g_pd3dDevice->SetRenderState(D3DRS_ALPHABLENDENABLE,TRUE);
g_pd3dDevice->SetRenderState(D3DRS_SRCBLEND,D3DBLEND_ONE);
g_pd3dDevice->SetRenderState(D3DRS_DESTBLEND,D3DBLEND_ONE);

D3DXMATRIXA16 matWorld1,matWorld2;
D3DXMatrixTranslation( &matWorld1,(float)(int)sx+2.0f,
   (float)(int)sy+2.5f,(float)(int)sz+2.0f);
D3DXMatrixScaling( &matWorld2,1.0f,1.0f,1.0f);
matWorld2 *= matWorld1;
g_pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld2 );
RenderModel(selectBox);

//アルファブレンディングの無効化
```

```
g_pd3dDevice->SetRenderState(D3DRS_ALPHABLENDENABLE,FALSE);

//文字の表示
g_ptextsprite->Begin(D3DXSPRITE_ALPHABLEND|D3DXSPRITE_SORT_TEXTURE);
LPCWSTR piece;
switch(board[(int)sy+2][(int)sz+2][(int)sx+2]){
case 1:   piece = _T("W_K");break;
case 2:   piece = _T("W_Q");break;
case 3:   piece = _T("W_U");break;
case 4:   piece = _T("W_R");break;
case 5:   piece = _T("W_B");break;
case 6:   piece = _T("W_N");break;
case 7:   piece = _T("W_P");break;
case 8:   piece = _T("B_K");break;
case 9:   piece = _T("B_Q");break;
case 10:piece = _T("B_U");break;
case 11:piece = _T("B_R");break;
case 12:piece = _T("B_B");break;
case 13:piece = _T("B_N");break;
case 14:piece = _T("B_P");break;
default:piece = _T("NONE");break;
}
RECT rc={0,400,640,520};
RECT rc1={0,360,640,520};
if(currentMode==1){
   if(currentTurn==T_W){
      g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
         _T("白のターンです.\n動かすコマを選んでください."),-1,&rc,
         DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
      g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
         piece,-1,&rc1,DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
   }else if(currentTurn==T_B){
      g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
         _T("黒のターンです.\n動かすコマを選んでください."),-1,&rc,
         DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
      g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
         piece,-1,&rc1,DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
   }
}else if(currentMode==2){
   g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
```

```
      _T("移動するマスを選んでください."),-1,&rc,
      DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
}else if(currentMode==3){
   g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
      _T("昇格できます. Q N R B U P"),-1,&rc,
      DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
   if(currentTurn==T_B){
      if( keys[DIK_Q]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=2;currentMode=1;}
      if( keys[DIK_N]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=6;currentMode=1;}
      if( keys[DIK_R]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=4;currentMode=1;}
      if( keys[DIK_B]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=5;currentMode=1;}
      if( keys[DIK_U]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=3;currentMode=1;}
      if( keys[DIK_P]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=7;currentMode=1;}

   }else{
      if( keys[DIK_Q]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=9;currentMode=1;}
      if( keys[DIK_N]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=13;currentMode=1;}
      if( keys[DIK_R]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=11;currentMode=1;}
      if( keys[DIK_B]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=12;currentMode=1;}
      if( keys[DIK_U]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=10;currentMode=1;}
      if( keys[DIK_P]&0x80 ){board[(int)sy2][(int)sz2][(int)sx2]=14;currentMode=1;}
   }
}

//チェックされるならば文字を表示して処理を戻す
if(checkCall){
   RECT rc2={0,0,0,0};
   g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
      _T("チェックされてしまいます."),-1,&rc2,
      DT_NOCLIP,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
}
//ゲーム終了状態ならば Enter キーで再スタート
if(restartCall){
   g_ptextsprite->Begin(D3DXSPRITE_ALPHABLEND|D3DXSPRITE_SORT_TEXTURE);
   RECT rc={0,430,640,520};
   g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
      _T("Enter で再スタート."),-1,&rc,DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
   g_ptextsprite->End();
}
//チェックメイトならば勝者を表示し終了
```

```
    if(checkmateCall){
        g_ptextsprite->Begin(D3DXSPRITE_ALPHABLEND|D3DXSPRITE_SORT_TEXTURE);
        RECT rc={0,400,640,520};
        g_pxfonts[hgamefont]->DrawTextW(g_ptextsprite,
            _T("チェックメイト."),-1,&rc,DT_LEFT,D3DCOLOR_COLORVALUE(1.0f,0,0,1.0f));
        g_ptextsprite->End();
    }
    g_ptextsprite->End();

    //リスタート
    if(restartCall&&(keys[DIK_RETURN]&0x80)){
        reset();
        checkmateCall=false;
        currentMode=1;
        currentTurn=T_W;
        restartCall=false;
    }
    if(isTimerGoal(0))checkCall=false;
    selectButton=cancelButton=false;
}

/**
 * 描画を行う関数、内部で GameMain 関数を呼び出す
 * @param void
 * @return void
 */
void Render(){
    g_pd3dDevice->Clear( 0, NULL, D3DCLEAR_TARGET|D3DCLEAR_ZBUFFER,
                         D3DCOLOR_XRGB(128,128,128), 1.0f, 0 );

    if( SUCCEEDED( g_pd3dDevice->BeginScene() ) ){
      GameMain();
      g_pd3dDevice->EndScene();
    }
    g_pd3dDevice->Present( NULL, NULL, NULL, NULL );
}

/**
 * 光源・カメラの初期位置・射影変換の設定を行う関数
 * @param void
```

```
 * @return void
 */
void SetViews(){
   //環境光の設定
   g_pd3dDevice->SetRenderState( D3DRS_AMBIENT,
      D3DCOLOR_COLORVALUE(1.0f,1.0f,1.0f,1.0f) );
   //ビュー変換
   D3DXVECTOR3 vEyePt( 2.0f, 2.0f,12.0f );
   D3DXVECTOR3 vLookatPt( 2.0f, 2.0f, 2.0f );
   D3DXVECTOR3 vUpVec( 0.0f, 1.0f, 0.0f );
   D3DXMATRIXA16 matView;
   D3DXMatrixLookAtLH( &matView, &vEyePt, &vLookatPt, &vUpVec );
   g_pd3dDevice->SetTransform( D3DTS_VIEW, &matView );
   //射影変換
   D3DXMATRIXA16 matProj;
   D3DXMatrixPerspectiveFovLH( &matProj, D3DX_PI/4, g_aspect, 1.0f, 100.0f );
   g_pd3dDevice->SetTransform( D3DTS_PROJECTION, &matProj );
}

/**
 * モデルのロードを行う関数
 * @param void
 * @return HRESULT:関数の呼び出しの詳細情報、上位 1bit が 1 なら失敗、0 なら成功
 */
HRESULT LoadModels(){
   ZeroMemory(&modelindex,sizeof(int)*20);
   modelindex[1] = LoadModel(_T("W_K.x"));
   if(modelindex[1]==-1) return E_FAIL;
   modelindex[2] = LoadModel(_T("W_Q.x"));
   if(modelindex[2]==-1) return E_FAIL;
   modelindex[3] = LoadModel(_T("W_N.x"));
   if(modelindex[3]==-1) return E_FAIL;
   modelindex[4] = LoadModel(_T("W_R.x"));
   if(modelindex[4]==-1) return E_FAIL;
   modelindex[5] = LoadModel(_T("W_B.x"));
   if(modelindex[5]==-1) return E_FAIL;
   modelindex[6] = LoadModel(_T("W_U.x"));
   if(modelindex[6]==-1) return E_FAIL;
   modelindex[7] = LoadModel(_T("W_P.x"));
   if(modelindex[7]==-1) return E_FAIL;
```

```
    modelindex[8] = LoadModel(_T("B_K.x"));
    if(modelindex[8]==-1) return E_FAIL;
    modelindex[9] = LoadModel(_T("B_Q.x"));
    if(modelindex[9]==-1) return E_FAIL;
    modelindex[10] = LoadModel(_T("B_N.x"));
    if(modelindex[10]==-1) return E_FAIL;
    modelindex[11] = LoadModel(_T("B_R.x"));
    if(modelindex[11]==-1) return E_FAIL;
    modelindex[12] = LoadModel(_T("B_B.x"));
    if(modelindex[12]==-1) return E_FAIL;
    modelindex[13] = LoadModel(_T("B_U.x"));
    if(modelindex[13]==-1) return E_FAIL;
    modelindex[14] = LoadModel(_T("B_P.x"));
    if(modelindex[14]==-1) return E_FAIL;

    MOV = LoadModel(_T("MOV.x"));
    if(MOV==-1) return E_FAIL;
    ATT = LoadModel(_T("ATT.x"));
    if(ATT==-1) return E_FAIL;

    selectBox = LoadModel(_T("select.x"));
    if(selectBox==-1) return E_FAIL;

    bar = LoadModel(_T("bar.x"));
    if(bar==-1) return E_FAIL;

    //フォント
    hgamefont = CreateGameFont(_T("MS ゴシック"),20,FW_BOLD);
    if(hgamefont==-1)return E_FAIL;

    return S_OK;
}

/**
 * WinMain 関数
 * @param hInst:現在のインスタンスのハンドル
 * @param HINSTANCE:以前のインスタンスのハンドル
 * @param LPSTR:コマンドライン
 * @param INT:表示状態
 * @return INT:終了時のステータス
```

```
 */
INT WINAPI WinMain( HINSTANCE hInst, HINSTANCE, LPSTR, INT ){
   ShowCursor( FALSE );   //カーソルを非表示
   if( SUCCEEDED( InitD3DWindow(_T("Raumschach(終了は Alt+F4)"), 640, 480) ) ){
      if( FAILED( LoadModels() ) ) return 0;   //モデルのロード
      SetViews();            //環境光・ビュー・射影変換の設定
      reset();   //board をリセット

      //メッセージループ
      lasttime = timeGetTime();   //ループ開始直前の時間を計測
      MSG msg = {0};
      while( msg.message!=WM_QUIT )
      {
         if( PeekMessage( &msg, NULL, 0U, 0U, PM_REMOVE ) ){
            TranslateMessage( &msg );
            DispatchMessage( &msg );
         } else{
            Render();
            DWORD curtime = timeGetTime();
            looptime = (float)(curtime - lasttime)/1000.0f;
            lasttime = curtime;
         }
      }
   }
   ShowCursor( TRUE );   //カーソルを表示
   UnregisterClass( _T("D3D Window Class"), GetModuleHandle(NULL) );
   return 0;
}
```