DXライブラリを用いたC++でのゲーム制作

仲村　聖実

１．研究概要

オブジェクト指向であるC++を開発言語とし、DXライブラリと呼ばれる、DirectXをWindowsプログラム特有のイベント稼働型プログラミングを意識せずにゲーム開発が行えるライブラリを使用してオセロを開発する。

２．研究の具体的内容

できる限り素材とアルゴリズムを自分で作っていき、自力でゲームを作成していくことを目的とした為､ゲームを開発するに当たり、プログラム、グラフィック、サウンドの技術が必要となる。それらをまとめて説明していく。

（i）プログラム-ゲーム構造

ゲームを作るといっても、基本的な構造がなければ制作するのは難しい。そこで､「新・ゲームプログラミングの館」にあるゲーム構造を取り入れた。

これは、ゲーム全体の処理と個々のオブジェクトに以下の処理に分けてプログラミングする方法である。

・初期化処理
・計算処理
・描画処理
・終了処理

これを導入することにより、それぞれの処理単位に分けることができ管理・プログラムしやすくなり、どこで問題が発生しているかを処理単位で見つけることが可能となる。

具体的に説明すると、図１のようになる。

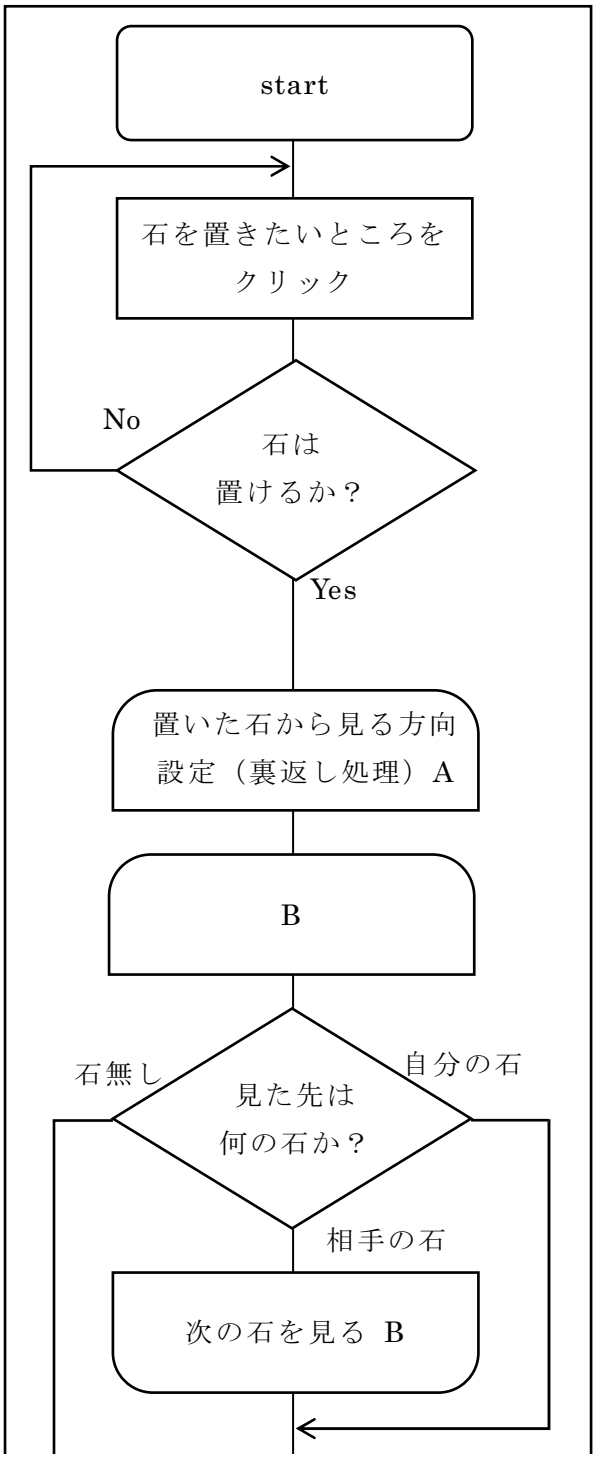
```
main(){
        初期化();
        while(1){
                if(計算())
                        break;
                描画();
        }
        終了();
}

初期化(){
        画像読込
        音声読込
        座標設定 etc...
}

計算(){
        主に動作内容
        例として、クリックされると
        敵キャラ画像が爆発画像に
        変わり、爆発音が鳴るように
        フラグを立てる
}

描画(){
        フラグにより画像表示させたり
        音を鳴らすなど
}

終了(){
        読み込んだ画像や音声を
        メモリ上から削除するなど
}
```

図１

(ii)プログラム-オセロ自体のアルゴリズム

現時点ではオセロのプログラムは完成していないが､アルゴリズムは既に完成している。

基本的な流れとしてはご存じの通り、黒のターン、白のターンの繰り返しとなり、板が全て埋まったらゲーム終了、数の多い方が勝利となる。自分のターンの時に、石を設置してから裏返るまでの大まかな流れ図を以下に示す。裏返す時に見る方向は上下左右斜めの８方向なので８回処理を行う。



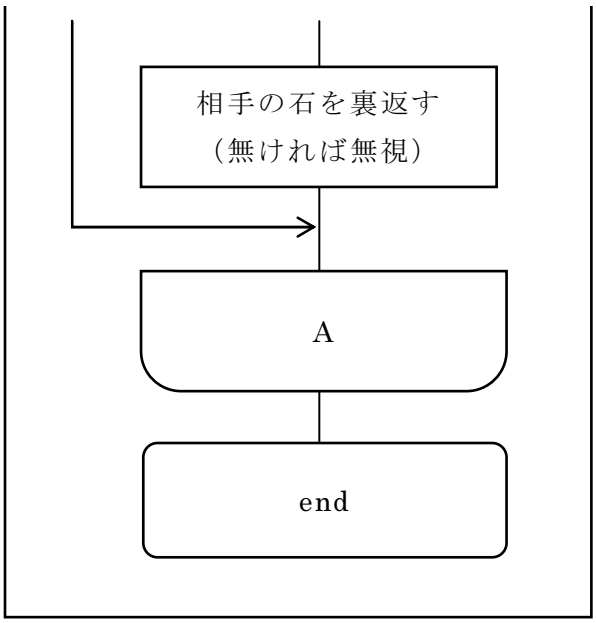


図２

一人でも遊べるようにコンピュータ対戦のアルゴリズムも完成している。

内容としては、石を取れる全てのパターンをシミュレートし、その中で一番多く石が取れたパターンを選択するもの。パターンが複数ある場合はその内からランダムに決められるようにしている。

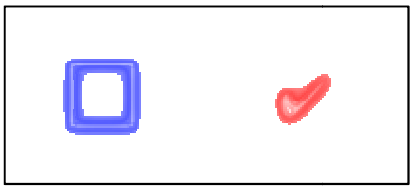
(iii) プログラム-ライブラリ

ゲームを効率的に作るためにライブラリを作成することにした。ライブラリとは複数の関数を一つにまとめたもので、共通して使える処理をまとめることにより作業の効率化が求められる。

ゲーム内のオブジェクト（例として、キャラクター、ボタン等）は画像データ、音声データ、数値データ、動作内容から成っていると考え、画像クラス、音声クラス、動作クラス（今回はマウスを使用するのでマウスの状態を得るクラスを作成した）を作成し、それらを継承して一つのクラスにまとめた。

画像クラスでは、画像の読込、表示位置となる座標データを主に使用するため、画像のハンドル値と座標データを確保している。画

像読込では、複数の画像を一つにまとめた画像（画像１参照）を分割読込できるようにもしたため、作業効率が良くなった。



画像１

また、ブレンド描画や画像の回転角度などの応用的使用にも対応できるようにした（画像２参照）。

```
//基本的必要データ
int  mx,             //x座標
     my,             //y座標
     *mHandle,       //ハンドル値（動的配列）
     mNumber;        //ハンドル値要素数
bool *mHandleCheck; //読込orハンドルチェック
                     //trueでハンドル値 falseでファイル読込

//補助的データ
int    *mBlend,      //ブレンド値
       mcx,          //x回転時中心座標
       mcy,          //y回転時中心座標
       msizex,       //xサイズ
       msizey,       //yサイズ
       mR,           //赤
       mG,           //緑
       mB;           //青
double mAngle,       //角度（ラジアン指定）
       mExtRate,     //全体拡大率（1.0で等倍）
       mExtRateX,    //x拡大率
       mExtRateY;    //y拡大率
```

画像２

音声クラスでは、音声の読込を行う。DX ライブラリの使用により、音量、パン、周波数の変更が可能なので、それらにも対応できるようにした。

画像クラスと音声クラスでは非同期読込に対応できるようにしている。非同期読込とは、メディアデータを読込完了するまで次の処理を行わない同期読込に対して、メディアデータを読込中でも次の処理を行うことである。簡単に言えば、ローディング画面を作る為にしている。アルゴリズムとしては、図３のようになる。

```
Load{
画像のファイル名や、分割読込の時は
分割数などのデータを一時的に保存する
}

Set(計算処理関数){
if(読込完了していたら)
        計算処理
else{
if(まだ画像読込を開始してないなら)
        画像を読み込む
else{
        switch(読込状況){
        case:読込完了
                読み込んだデータを保存
        case:読込中
                読込続行
        case:読込失敗
                エラーを返す
        }
}}
}
```

図３

動作クラスでは、マウスの状態を取得し、取得したデータによりフラグを変更する。フラグには、クリックされたか、対象物上にマウスポインタがあるか、動作可能か、動作不可か、動作したかの５つのフラグを使用している。これにより、対象物上以外でドラッグし、そのままポインタを対象物上に移動してドロップしても動作させないという基本的な動作を簡単に実装できるようになった。

これら３つのクラスを継承したクラスでは、クラス自体を変更することなくクラスを使用できるようにする為、仮引数にフラグを持たせる為符号無しの long int のポインタ型としたポインタ関数を持たせた。また、クラス内

にデータを持たせるように全ての型に動的配列用のポインタを持たせた。これらにより、汎用的にプログラムを組むことが可能となった。
以下に４つのクラスをまとめたものを図４に示す。

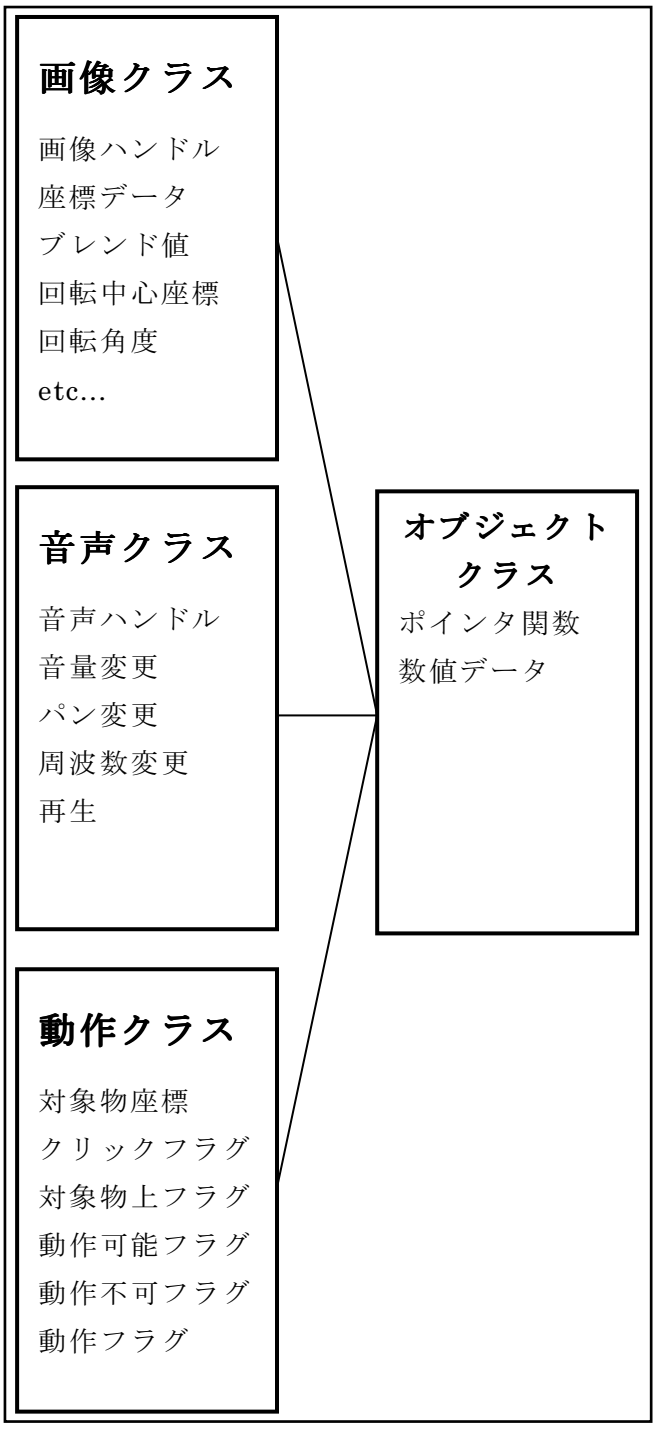


図４

(iiii)　プログラム-シーン管理

ゲーム作成を作りやすく、分かりやすくするためにシーン管理をするアルゴリズムを導入した。タイトル画面やゲーム画面などを一つのシーンとして扱い、シーンごとに集中してプログラミングすることが可能となる。

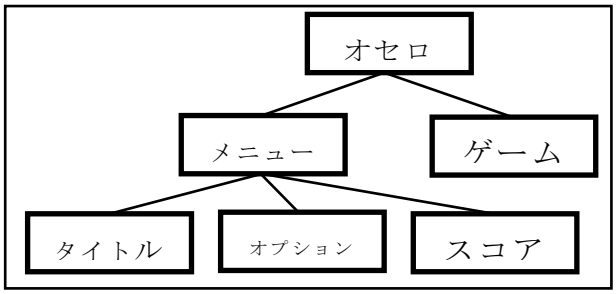


図５

上記の図５のように､オセロ(ゲーム本体)はメニューとゲームを管理し、メニューはタイトル、オプション、スコアを管理するようなアルゴリズムとなっている。このアルゴリズムは「新・ゲームプログラミングの館」にあるプログラムを改造して使用している。それぞれのシーンに必要なメディアデータを読み込ますことも容易になり、メモリの節約も行えた。シーン間のメッセージのやり取りは後述するデータ管理にて行う。

(iiiii)　プログラム-データ管理

効果音の音量やオプションで決めた設定など、どのシーンでも使用したいデータは存在する。

そこで、全てのシーンで共通データを扱えるように、BGM や効果音、設定値をそれぞれクラス化してそれらを一つにまとめたクラスを作成し、そのクラスのポインタを全てのシーンに持たせた。これによって容易に共通データを使用することができる。このデータには効果音の音量だけでなく、fps 制御、共通メディアデータ、ゲームモード、スコアなどを入れている。使用イメージとしては次ページの図６のようになる。

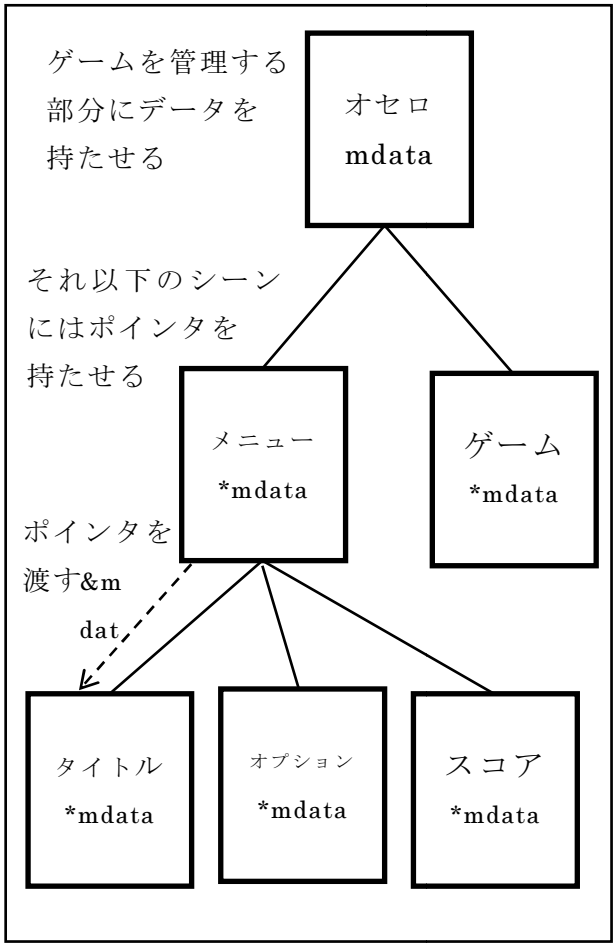


図６

fps 制御とは、ゲームスピードをどのディスプレイでも同じスピードで動作させる為のものである。仮に、90Hz のディスプレイでゲームを動かすと、ゲーム自体は 60 フレームで動く為 1.5 倍のスピードでゲームが進行してしまう。

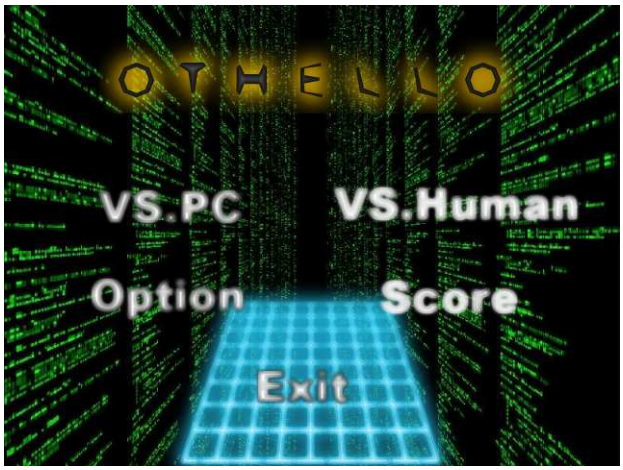
これを防ぐために導入している。

共通メディアデータとは、同じメディアデータを使用する時に予めデータ管理側で読込しておき、そのハンドル値を渡すもの。これにより同じメディアデータを読み込む必要が無いのでメモリの節約となる。

（iiiiii）　グラフィック

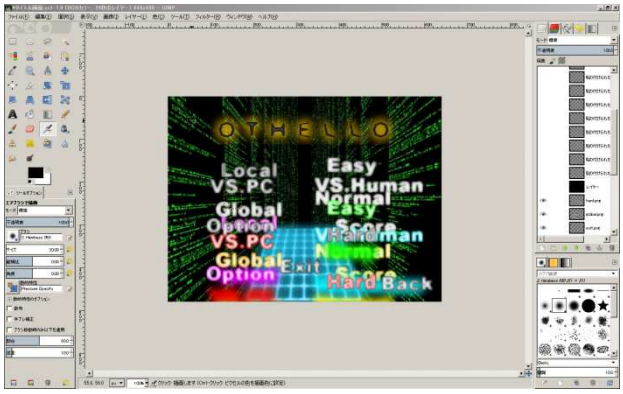
「近未来風オセロ」をコンセプトとして画像制作を行った。画像編集ソフトは GIMP を使用した。GIMP はフリーでありながらも高機能・高性能な加工が可能となるソフトである。

以下はタイトル画面の画像である。



画像３

プログラムでは座標を入れて画像を表示させるが、画像編集ソフトのように画面を確認しながら画像の座標を決めることは不可能なので GIMP で座標を確認しながらプログラムを行った。以下の画像参照。



画像４

（iiiiiii）　サウンド

BGM や効果音は当初は自作する予定だったが、時間の都合により行えなかった。その為、インターネットを通じて素材をお借りしている。

当初は、Domino と呼ばれる作曲ソフトと ProteusVX と呼ばれるソフトウェア音源を使用して作曲する予定だった。

Domino はピアノロール形式で作曲することが出来るので五線譜が読めない場合でも作曲をすることが可能なフリーソフト。音量やパンなどを直感的に変更することも可能なので表現の幅は広い。

ProteusVXは元々有料の音源だったが現在はフリー化されているソフトウェア音源。1000個以上の音色を選択でき、またプリセットを元に音色作りも可能。

音声ファイルの拡張子はwavだと容量が大きい為、特許の存在しないoggに変換して使用した（mp3にすると特許の問題が発生するのでゲーム作成には使われない)。BGMを再生する時はメモリ節約の為にストリーミング再生を行った。ストリーミング再生とは、読込中と同時に再生することである。

３．研究のまとめ

当初はほとんどの素材を自作する予定だったが、プログラム面でのライブラリの作成に非常に時間がかかってしまい、それどころでは無くなった為にサウンド関係は素材をお借りすることとなってしまった。その代わり、ライブラリの作成方法や動的配列の使い方、テンプレートなど、ライブラリを作成しなければ触れることの無かった知識に触れることができ、その面ではしっかりと学習ができた。

C++を使用するのは初めてとなるので２年生の３学期くらいから学習をしていた。そのおかげで言語につまずくことはほとんどなかった。また、C++の特徴であるオブジェクト指向を理解することができ、オブジェクト指向を使ったアルゴリズムを考えることも可能になった。ただ、スペルミスや何でもないエラーを頻繁に起こしていたために（時々深刻なエラーも）エラーの修正に１週間かかることもあり、プログラムの作業の半分はエラーの修正にかかってしまった。今後プログラマーとして働く立場としては反省すべき点である。

グラフィックはほとんど自作だが、画像がワンパターン化してしまっているのでもう一工夫するべきと反省した。ゲームはどう魅せるかも勝負になると思うので、グラフィック技術の向上を目指していくようにする。

サウンドは作曲などしていく予定だったが、環境だけ整っていて作成することができなかった。主にしたのは変換とカットぐらいなので、こちらの方面もしっかりと学習していく。

ゲーム自体はライブラリの完成が大幅に遅れたため、これから実装していく形となる。プレゼン発表までに完成を目指し、今後努力していく。

４．感想

結論から言うと、ゲーム作成をなめていた。

コマンドプロンプト上で動作するオセロやテトリスを作成したことがあるが、その時は一週間程度で作成できた。順調に出来るだろうと計画していたが、予想とは逆方向に進んでいき、計画通りに全く進まなかった。そうなったのはゲームプログラミングの効率化を図る為に導入したシーン管理とライブラリ作成が主な原因だった。シーン管理は使用・改造する為にアルゴリズムを解読したが、当時はC++への理解が十分でなかったために完全に理解するのに一か月もかかってしまった。導入するのにも慣れないエラーの解決に時間がかかった。ライブラリは、どうすれば効率良く、汎用的に使えるのか考えるのに時間を使い、また動的配列や非同期読込の対策に非常に時間を使った。事前調査もほとんどせずに計画を立ててしまったことに反省した。これから巻き返しを図る。

最後に、この課題研究を通して様々な技術を取得したので今後の自分の活動の為にこの技術を活かしていけるようさらに努力していく。

５．参考文献

(1)ロベールのC++入門講座
著：ロベール
(2)新・ゲームプログラミングの館
http://dixq.net/g/
(3)ＤＸライブラリ置き場
http://homepage2.nifty.com/natupaji/DxLib/