

第1回 DX ライブラリについて

本日からソフトゼミBを開始していきます。ソフトゼミBではソフトゼミAで学んだC言語の知識を元に、実際にゲームを作成していき、ゲームプログラミングの基本的な部分を学んでいきます。

ゲームをプログラミングする際、1から自分でプログラミングしていくのは非常に困難です。今までは入出力をコマンドプロンプト上で行うプログラムを作ってきました。これらは基本的にどのOSでもきちんとプログラムしておけばコンパイラをOSに合わせるだけできちんと同じように表示されます。しかし、ゲームはそうは行きません。各OSごとに異なる文化があるため、ウィンドウを作成したり、画面上に画像を表示するだけでも一苦労です。

今回は、そのような、ゲーム自体のプログラミングから離れた煩わしい部分を抜きで、Windowsプログラムだと意識せずに済むように「DX ライブラリ」というゲーム制作用ライブラリを用いてシューティングゲームを作ります。

○ ソフトウェアのダウンロード

DX ライブラリを使用するには、以下のソフトをインストールする必要があります。

- ・VisualC++ 2010 Express Edition

<http://www.microsoft.com/japan/msdn/vstudio/express/>

このVisual C++ 2010 Express の部分の少ししたの「Web インストール(ダウンロード)」からダウンロードできます。Windows Vista、7 の人は上記のものがおすすめですが、XP ユーザーの場合は同2008 がオススメです。上記のページの右下の「過去のバージョン」から、Visual C++ 2008 Express Edition のロゴの右下の「Web インストール(ダウンロード)」をクリックすることでインストーラをダウンロードできます。

上記のソフトウェアはDX ライブラリを使うことができるソフトウェアの中で、無償提供されているものです。他にも、有償のソフトウェアもありますが、今回は使用しません。

また、これに加え、以下のソフトを使用します。

- ・DX ライブラリ VisualC++用

<http://homepage2.nifty.com/natupaji/DxLib/index.html>

「DX ライブラリのダウンロード」から、ダウンロードページに跳ぶことができます。上のVisualC++用をダウンロードしましょう。

○ DX ライブラリの導入

DX ライブラリを使う上での手順は以下の4ステップにまとめられます。

- ・DX ライブラリの入っているディレクトリの指定
- ・プロジェクトの文字セットの変更
- ・プロジェクトのマルチコアにおける動作の変更

これが結構厄介です。DX ライブラリのマニュアルの「DX ライブラリの使い方」に詳しく載っているのでそれを参照しながら設定していきましょう。ただし今回はプログラムのソースコードのファイル名を「DrawPixel.cpp」ではなく「main.cpp」とします。

○ DX ライブラリ実践

以下のプログラムを先程作成したmain.cppに写し、実行してください。プログラムは2ページに渡っているので注意してください。また、「int WINAPI WinMain(中略) {」は1行に書いてください。

```
// インクルード
// DX ライブラリ関数をインクルード
#include "DxLib.h"

// メイン関数
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
```

```

nCmdShow ){
    ChangeWindowMode( TRUE ); // ウィンドウモードにする

    if( DxLib_Init() == -1 ){
        // DX ライブラリの初期化に失敗した場合、即座に終了させる
        return( -1 );
    }

    DrawBox( 0, 0, 32, 32, 0xffffffff, TRUE ); // 四角形を描画
    WaitKey(); // キーが押されるまで待機

    DxLib_End(); // DX ライブラリを終了
    return( 0 );
}

```

ウィンドウ上部のメニューから、「デバッグ」->「デバッグ開始」を押すと、コンパイルし実行します。コンパイルが無事終了し、ウィンドウが表示され、画面左上に白い四角形が描画されれば成功です。デバッグとはプログラムが正常動作するかどうかの試験を行うことです。上記のプログラムは「main_01a.cpp」に書かれています。

さて、実際にゲームを作っていきます。先ほどのプログラムから、DrawBox 関数の行と、WaitKey 関数の行を消去し、その部分に以下のように追記してください。

```

// 今のところはおまじない(「DX ライブラリミニテクニック 裏画面のすすめ」参照)
SetDrawScreen( DX_SCREEN_BACK );
// 実際にゲームのループに入っていく
Game();

```

前者の関数については気にしないでください。後者の関数はまだ作成されていない、ゲームを動かす関数です。そのため、赤く波線が表示されます。最初のインクルードの後にプロトタイプ宣言をして WinMain 関数よりも後に Game 関数を書く(推奨)か、WinMain 関数よりも前に Game 関数を書いてきましょう。内容は以下の通りです。

```

// ゲーム内容
int Game( void ){
    // ループ突入
    while( ProcessMessage() != -1 ){
        // 基本的な流れは 入力->移動->判定->描画
        // (入力から入力結果反映までを最短にする)
        /*****
            ここから入力
            *****/
        // まだプログラムしないよ
        /*****
            ここまで入力
            *****/

        /*****
            ここから移動
            *****/
        // まだプログラムしないよ
        /*****
            ここまで移動
            *****/
    }
}

```

```

/*****
    ここから判定
    *****/
// まだプログラムしないよ
/*****

    ここまで判定
    *****/

/*****
    ここから描画
    *****/
ClearDrawScreen(); // 一度画面を消去
DrawPlayer();      // 自機描画
ScreenFlip();      // DX ライブラリミニテクニク 裏画面のすすめ参照
/*****

    ここまで描画
    *****/
}
// エラー発生時はゲーム終了処理
return( -1 );
}

```

コメントアウトばかりですが、我慢して書いていきましょう。こうすることで後で見やすくなるはず。上のプログラムから分かるように、ゲームプログラムの基本的な流れは、コントローラ等からの入力の受け取り→画面上のオブジェクト(敵キャラクターや自機等の物体)の移動→当たり判定→描画、の繰り返しであることが分かります。この繰り返しをごく僅かな時間に何回も(基本的には毎秒60回)行うことで、滑らかにキャラクターが移動しているように見える、という仕掛けです。今回は、まだ敵や自機の情報を記録しておく部分を作成していないので、DrawPlayer 関数には画面中心に円を描画するプログラムを書いていきます。先程と同様に、プロトタイプ宣言をインクルードの直後で行い、DrawPlayer 関数を main 関数よりも後に書く(推奨)か、main 関数よりも前に DrawPlayer 関数を書いていきましょう。内容は以下の通り。

```

// 自機の描画
void DrawPlayer( void ){
    DrawCircle( 320 , 240 , 8 , GetColor( 0 , 255 , 0 ) , TRUE );
}

```

これで、画面の中心に半径8ドットの緑色の円が描画されます。この時点でのプログラムは「main_01.cpp」に書かれています。

さて、今回使ったDXライブラリの関数の機能について、おさらいしていきましょう。

ChangeWindowMode 関数は画面のモードを切り替える関数です。全画面か、ウィンドウモードかの切り替えを行うことができます。この関数は特別に、DXライブラリの初期化を行う前に使用することができるDXライブラリの関数の1つです。

DxLib_Init 関数はDXライブラリを初期化する関数です。初期化に成功すれば0を、失敗した場合は-1を返り値として返します。-1が帰ってきた場合はプログラムをただちに終了しなければなりません。

DrawBox 関数は四角形を描画する関数です。画面左上を(0, 0)とし、xの正の方向を右、yの正の方向を下とした座標系において、第1、2引数を四角形の左上の頂点座標、第3、4引数を右下+1の頂点座標とする長方形を描画します。第5引数で色、第6引数で内側を塗りつぶすか否か(TRUEかFALSEで指定)を指定します。色については、16進数で左から順に2桁が赤の強さ、2桁が緑の強さ、2桁が青の強さです。0で最も弱く、255(FF)で最も強くなります。

GetColor 関数は上記における色を簡単に扱えるようにする関数です。第1、2、3引数にそれぞれ赤、

緑、青の強さを 256 段階(0～255)で指定すると、返り値として先程の関数の説明で説明したような値を返します。

WaitKey 関数はその名の通り、キー入力を受け付けるまでプログラムの動作を停止する関数です。

DxLib_End 関数はDX ライブラリを終了する関数です。DX ライブラリの初期化に成功した場合は、以後、プログラムを終了する場合、必ずこの関数を実行してから終了するようにしてください。また、この関数を使用してもプログラムが終了するわけではありません。WinMain 関数で return するか、exit 関数を実行する必要があります。また、これより以後にDX ライブラリの関数を使用してはいけません。

SetDrawScreen 関数は描画先のスクリーンを変更する関数です。ここでは詳しく説明はしません。知りたい方はDX ライブラリミニテクニック 裏画面のすすめを参照してください。

ProcessMessage 関数はWindows に様々な通知を行う関数です。1/60 秒に1 度くらいこの関数を実行しなくてはなりません。つまり、1 ループに1 回実行するようにしてやってください。また、この関数が-1 を返り値としてかえた場合、エラーがおきているのでループを抜けて直ちにプログラムを終了する必要があります。この時、必ずDxLib_End でライブラリ使用を終了してからプログラムを終了するようにプログラムしてください。

ClearDrawScreen 関数は画面を初期化する関数です。これをしないと、以前の画面の上から現在の画面が描画され、ひどいことになります。次回以降実際にやってみれば、どのようにひどいことになるのか確認できるでしょう。

ScreenFlip 関数は(今回のプログラムでは) 描画結果を画面に反映する関数です。ここでは詳しく説明はしませんが、知りたい方はDX ライブラリミニテクニック 裏画面のすすめを参照してください。

DrawCircle 関数はその名の通り円を描画する関数です。第1、2 引数に中心のXY 座標、第3 引数に半径、第4 引数に色、第5 引数に塗りつぶすか否か(TRUE かFALSE)を指定します。

上記の説明は今の時点で知っておくべきことのみを記述しています。DX ライブラリの関数の詳細についてはリファレンスマニュアルを必ず参照しましょう。

今回のゼミは以上です。お疲れ様でした。今回製作したプログラムは次回以降も使用していきますので、持参してください。