# 第 1 回　DX ライブラリの導入

今回からはソフトゼミ B です。ソフトゼミ B ではゼミ A で学んだ C 言語を元に、実際にアクションゲームを作っていきます。

初めからゲームを作るのは難しいので、ライブラリという関数の集まりを使って作っていきます。今回は DX ライブラリというゲーム用のライブラリを使ってアクションゲームを作ります。

## DX ライブラリの導入

DX ライブラリを使うには以下のソフトウェアをインストールする必要があります。

・ViisualC++ 2008 Express Edition

・VisualC++用 DX ライブラリ

これらはそれぞれ以下のアドレスから入手できます。

VisualC++ 2008 Express Edition

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

ここのページの下の方。VisualC++2008 から Web インストール。

似たような名前が多いので注意

VisualC++用 DX ライブラリ

http://homepage2.nifty.com/natupaji/DxLib/dxdload.html

VisualC++用をダウンロード

## DX ライブラリの設定を行なう

ダウンロードが終わったら、VisualC++のインストールをします。すごく時間がかかりますので、お茶でも飲んでまったりしておいてください(1 時間くらい)。

インストールが終わったら、

スタート＞プログラム＞VisualC++9.0 Express Edition>Microsoft Visual C++2008 Express Edition

から VisualC++を起動します。

無事、VisualC++が起動できたら、次に DX ライブラリの導入を行ないます。

・VisualC++ 2008 Express Edition のメニューの『ツール』→『オプション』を選びます。

・オプションウインドウの左側から

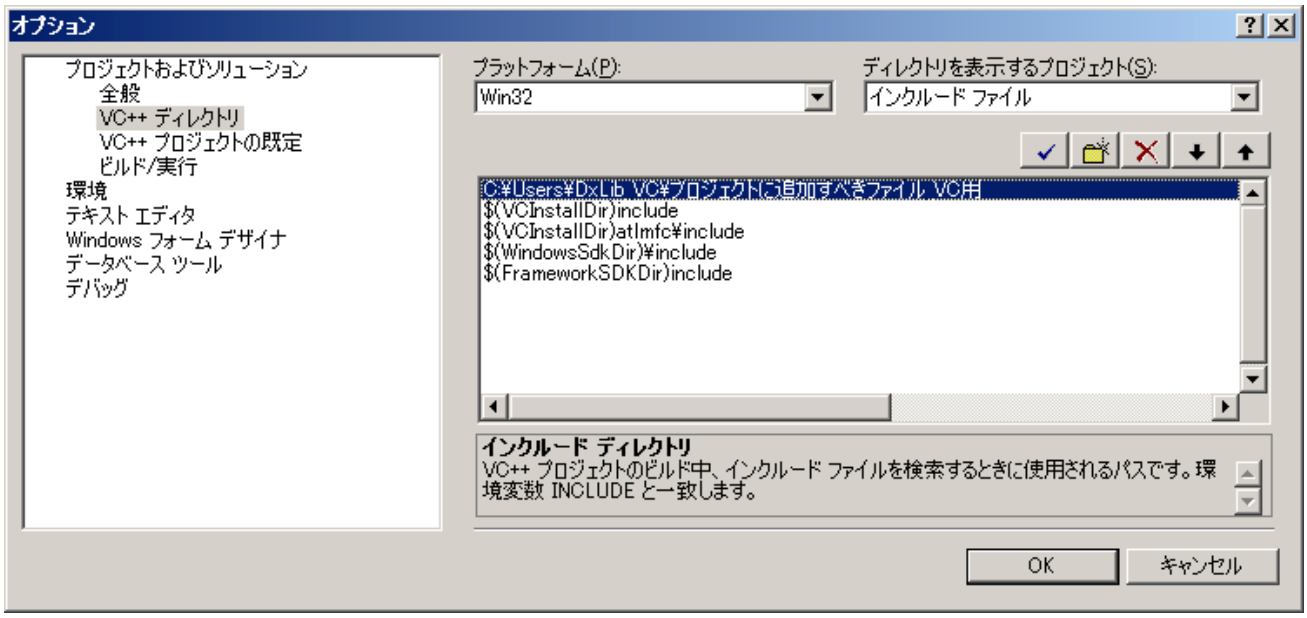
『プロジェクトおよびソリューション』→『VC++ ディレクトリ』を選びます。

・右側の『ディレクトリを表示するプロジェクト』から『インクルードファイル』を選びます。

・ディレクトリリストにＤＸライブラリのパッケージ内に入っている

『プロジェクトに追加すべきファイル_VC 用』フォルダのパスを追加します。

・『ディレクトリを表示するプロジェクト』を今度は『ライブラリファイル』にします。
・ディレクトリリストに□と同じく
　『プロジェクトに追加すべきファイル_VC 用』フォルダのパスを追加します。
・『ＯＫ』ボタンを押して設定完了です。




今、やってもらった作業が Visual Studio に DX ライブラリを使えるようにする設定です。これは Visual Studio をインストールした時にだけやればいいです。

次にプロジェクトを作成します。プロジェクトというのはソースファイルをまとめて管理するファイルとかそういう感じです。

- VisualC++ 2008 Express Edition のメニューの
  『ファイル』→『新規作成』→『プロジェクト』を選びます
- 『プロジェクトの種類』の項目から『VisualC++』→『Win32』を選びます
- 『テンプレート』の項目から『Win32 プロジェクト』を選びます
- 次にプロジェクトを作成するフォルダパスを『場所』で設定します
- 次にプロジェクトの名前を『プロジェクト名』に入力します(なんでも構いません)。
- すると『Win32 アプリケーションウィザード』のウィンドウが出るので
  ウィンドウの左側にある『アプリケーションの設定』を選択します
- そしたら右側になにやらたくさんの項目が出てきますが、『追加オプション』欄の
  『空のプロジェクト』にだけチェックを入れて下の『完了』を選択します。

さて、プロジェクトが作成できました。そろそろ疲れてきましたが、まだまだあります。

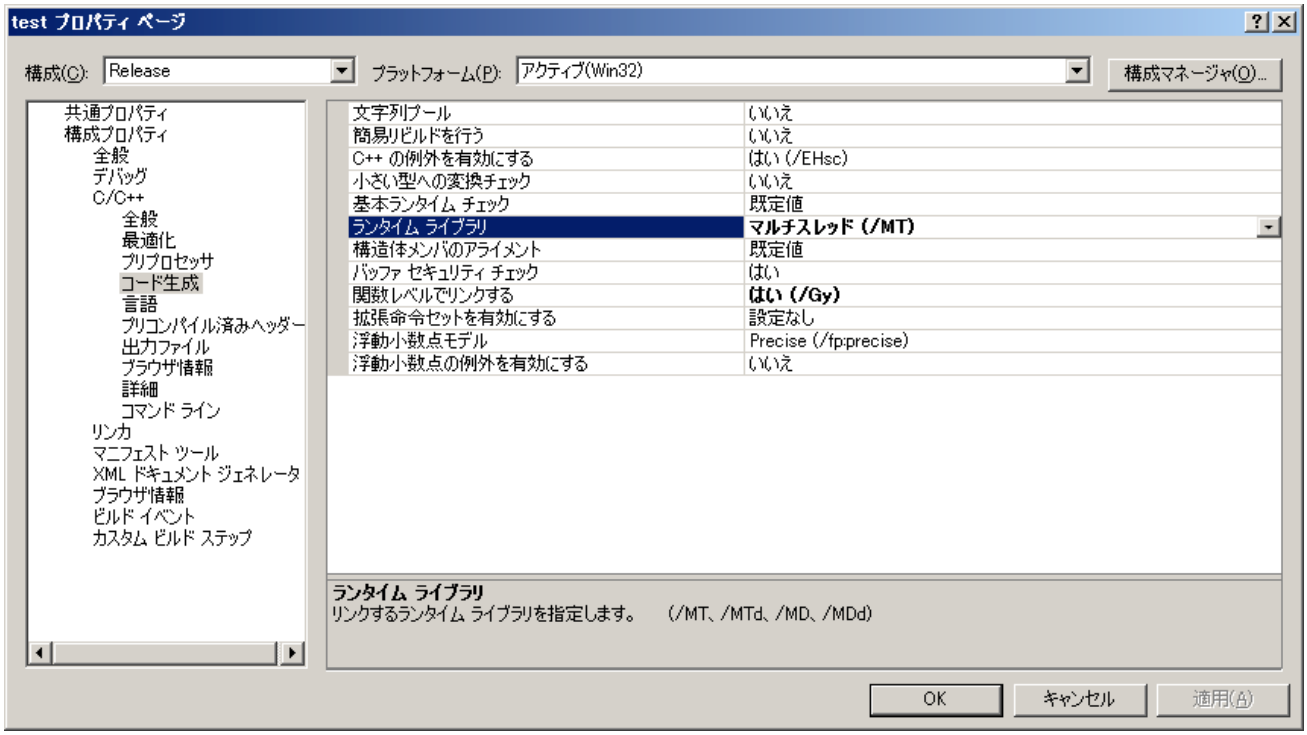
- Visualc++ 2008 ExpressEdition のメニューから
  『プロジェクト』→『新しい項目の追加』を選び、『新しい項目の追加』を開きます。

・　ウィンドウ左側の『カテゴリ』から『VisualC++』を選び、右の『テンプレート』から『C++ファイル(cpp)』を選び、下の『ファイル名』欄にファイル名を入力し(今回は main.cpp にします)、『追加』ボタンを押します。

これで空のソースファイルは作成されました。.cpp とは C++ファイルという意味です。

飽きてきましたが、まだあります。次にプロジェクトの設定を行います。

・　VisualC++ 2008 Express Edition のメニューの『プロジェクト』→『～～のプロパティ』を選択します。
・　プロパティダイアログが開いたら、ダイアログの左のリストから『構成プロパティ』→『全般』を選びます。
・　ダイアログ右側に表示されている『文字セット』の項目を『マルチバイトセットを使用する』に変更します。
・　左側のリストから今度は『構成プロパティ』→『C/C++』→『コード生成』を選びます。
・　ダイアログ右側に表示されている『ランタイムライブラリ』の項目を『マルチスレッド　デバッグ(/MTd)』に変更します。
・　次にダイアログ左上に表示されている『構成』の項目を『Release』に変更します。
・　ダイアログ右側に表示されている『ランタイム ライブラリ』の項目を今度は『マルチスレッド(/MT)』に変更します。
・　最後にダイアログの下の方にある『OK』を押してダイアログを閉じます。



やっとプログラムを実際に書く準備が整いました。それでは実際に main.cpp に次のようなプログラムを書いてみましょう。

```
#include “DxLib.h”
//DX ライブラリの関数をインクルードする

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すると即座に終了する
                return -1;
        }
        //ここまでおまじない。わー

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

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

これを先ほどのファイルに書いてください。

VisualStudio でのコンパイルの仕方はメニューから『デバッグ』→『デバッグ開始』を押すとコンパイルと実行をしてくれます。それで『デバッグ開始』を押し、無事にウィンドウが作られ四角形が表示されたら OK です。ちなみにデバッグとはプログラムが動くかテスト動作させるとかそんな意味です。

それではこれからゲームを作っていきます。上記のプログラムは DX ライブラリの動作の確認のためなのですが、そのまま流用します。とりあえず DrawBox と WaitKey は要らないので、その２行を消しておいてください。

次に、ゲームで使う変数を用意します。main.cpp の最初のところ(#include の次の行)に下記のプログラムを付け足してください(コメントアウトはご自由に)。

```
#define TIP_MAX_X 128
#define TIP_MAX_Y 32

#define TIP_HABA 32
#define TIP_TAKASA 32
//チップの幅と高さ
//チップって言うのはマリオでいうブロックです

#define JIKI_HABA 32
#define JIKI_TAKASA 64
//自機の幅と高さ
//自機の画像を変えた場合はここを修正すること
```

```
struct mono{
	int haba;
	int takasa;
	int x;
	int y;
	int flag;
	int handle;
};
//自機、及び敵に関する構造体
//haba、takasa はその物体の幅と高さ
//x、y はその中心座標
//handle は画像のハンドルをもらうための変数
struct mono jiki;

int tip[TIP_MAX_X+1][TIP_MAX_Y+1];
int tip_handle;

//tip[ ][ ]っていうのはチップの二次元配列
//配列数を+1 にしているのは色々と間違う可能性があるから冗長を持たせてみました
//tip_handle はチップの画像を読み込むための変数
```

なんだか一気に増えましたね。変数名をちゃんとしておかないと後が大変になります。

#define とは定数を文字として置いたものです。イメージとしては値が変更でいない変数みたいな感じです(厳密には違うけど)。struct mono はこれから自機や敵キャラなどの物を保存するための変数一式です。このあと速度などの変数もこの構造体に付け足してきます。

次にゲームの本体となる関数を書きます。

```
void g_loop( void){
	int i, j;

	jiki.handle = LoadGraph(“data/jiki.bmp”);
	//自機画像をロード
	jiki.x = 300;
	jiki.y = 200;
	//自機の初期座標セット

	tip_handle=LoadGraph(“data/tip.bmp”);
	//チップ画像のロード

	for( i = 0; i <= TIP_MAX_X; i++){
		for( j = 0; j <= TIP_MAX_Y; j++){
			tip[i][j] = 0;
		}
	}
	tip[1][1] = 1;
	tip[1][3] = 1;
	//for 文二重ループからこの辺までは第 2 回で説明します
```

```
	while( m_escape() >= 0){
		g_tip_draw();
		//チップの描画をする

		g_jiki();
		//自機に関することをまとめた関数

		ScreenFlip();
		//画面に描画する

		WaitTimer( 16);
		//待つ。ゲームは普通秒間 60 フレームなので 1/60 = 0.01666…
		//単位はミリ秒なので 16 が引数になる

		ClearDrawScreen();
		//画面をクリアにする
	}
}
```

```
int m_escape( void){
	//ProcessMessage が-1 のときと ESC キーを押したときに-1 を返す
	//ProcessMessage と ESC キーを同地にするために一つの関数に
	//つまり ESC キーで抜けられる箇所=ProcessMessage が入っているということ
	//ESC で抜けられない箇所で、×ボタンや Alt+F4 を押すと
	//タスクが残ってあれれ～？ってなる

	if( ProcessMessage() == -1) return -1;
	else if( CheckHitKey( KEY_INPUT_ESCAPE) == 1) return -1;
	//戻り値-1 でエラー。さっさと終了しろということ

	return 0;
}
```

まず、ゲーム本体となる g_loop という関数を作ります。ゲームは 1 フレームごとに座標を計算したり、敵を動かしたりします。その繰り返しによってゲームは動いて見えます。ScreenFlip という関数がありますが、これは 1 フレームを進めるようなものだと思っておいてください。WaitTimer というのは指定した時間だけ待つ関数です。60fps 前後にしたいので 16ms だけ待ちます(ちなみに今のパソコンはめっちゃ速いから計算の時間はほとんど考慮しなくていいよ)。

m_escape 関数は 1 フレームに呼び出してください。m_escape の中にある ProcessMessage という関数はエラーが起きたときに-1 を返します。エラーが起きたときはただちにプログラムを終了しなければなりません(いや、ホント)。ここでは g_loop の while 文の終了条件に使って、異常時はすぐに終了するプログラムにしています。

次に実際に画像を描画するプログラムを書きます。

```
void g_jiki( void){
        //自機に関する関数

        DrawGraph( jiki.x – JIKI_HABA / 2, jiki.y – JIKI_TAKASA / 2, jiki.handle,
        true);
        //自機を描画する
}
```

```
void g_tip_draw( void){
    //チップの描画をする関数
    int i;
    int j;
        for( i = 0; i <= TIP_MAX_X; i++){
            for( j = 0; j <= TIP_MAX_Y; j++){
                if( tip[i][j] == 1){
                    //本来ならば、画面内のチップしか描画をする必要はない
                    //が、面倒なので全部描画させます
                    //処理が追いつかない場合は自分で考えて

                    DrawGraph( i * TIP_HABA - TIP_HABA / 2, j * TIP_TAKASA –
                    TIP_TAKASA / 2, tip_handle, false);
                    //幅、及び高さの 1/2 を x、y から引いてるのは、
                    //画像の中心を x、y にするため。
                    }
            }
      }
}
```

DX ライブラリでは画像を表示するために、メモリ上に画像を読み込む関数と、描画する関数を使って画面に絵や図形を描画します。ここでは LoadGraph が画像をメモリの読み込む関数、DrawGraph が画面に描画する関数です。

DrawGraph 関数は x 座標、y 座標、画像ハンドル、透過の有無の 4 つを引数にとります。x 座標、y 座標は描画する画像の一番左上を基準とします。なので画像の中心を基準としたいので、画像の半分の大きさ分だけ左上側に移動させて描画させています。

チップは tip[i][j]という二次元配列であるので、for 文の二重ループにすることによりすべての tip 画像に対し if 文での判定を行い、描画するか、しないかを判定します。

さて、画像チップを描画する関数ができたのなら、実際に main 関数に組み込んで描画してみましょう。

```
#include “DxLib.h”

/*
中略
ここに先にやった struct とか int key_a とかの一式をコピペする
*/

void g_loop( void);
int m_escape( void);
```

```
void g_jiki( void);
void g_tip_draw( void);
//プロトタイプ宣言。関数の目次みたいなもん

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
	LPSTR lpCmdLine, int nCmdShow)
{
	ChangeWindowMode( true);
	if( DxLib_Init( ) == -1){
		return -1;
	}
	//ここまでおまじない

	g_loop( );
	//メインループへ

	DxLib_End( ) ;
	return 0;
}

//ここから下に先ほどの g_loop とか入れる
```

メイン文はこんな感じになります。今はほとんど何もありませんが、後々タイトル画面やオプション画面をつけることになるとそれなりに繁盛します。

さて、実際にここまで正しく打ち込めたのであれば、実際に実行してみましょう。
ここで素直にコンパイルが通れば幸せです。

あと DxLib_End という関数が最後にありますが、これは DX ライブラリを終了させる関数なので必ず入れてください。いれないとロードした画像データがメモリから解放されなくなったりしてシステムが不安定となってしまいます。