ＤＸライブラリ講座

① ＤＸライブラリとは

ＤＸライブラリはただプログラマーの方々が絵を画面に表示することや、効果音を鳴らす
ためだけに膨大な労力を費やさずにすむために存在するライブラリなのです。
(DX ライブラリの本家サイト(以下サイトと書きます)「どこまで出来るＤＸライブラリ」のページより抜粋)

ゲームを作るのに必要な画像表示・音楽再生・入力処理などの基本的な部分をいくつかのライブラリ関数で行ってくれます。
どこまでできるかはサイトの「どこまで出来るＤＸライブラリ」項などを読んでください。

ＤＸライブラリサイトのＵＲＬ→ http://homepage2.nifty.com/natupaji/DxLib/index.html

② ＤＸライブラリの導入

サイトにも書いてありますが一応。(前提として VisualStudio は入っているとします)

1. サイトからＤＸライブラリをダウンロードし、解凍(DxLib_VC というフォルダが出てきます。)
2. 出てきたフォルダを適当な箇所に移動（誤って削除したりしない場所がいいです)
3. VisualStudio を起動
4. 『ツール』→『オプション』→『プロジェクトおよびソリューション』→『VC++ ディレクトリ』
5. ウインドウ右上の『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』を選択
6. 『プロジェクトに追加すべきファイル_VC 用』フォルダ(DxLib_VC フォルダにあります)の場所を追加
7. 『ディレクトリを表示するプロジェクト』を『ライブラリファイル』に変更
8. 『プロジェクトに追加すべきファイル_VC 用』フォルダの場所を追加

以上でＤＸライブラリが使えるようになります。

③ ソースコード作成前にすること

それでは実際に作成…といきたいところですが､プロジェクトを作成する際に**毎回やらなければならない**ことがあります。

1. 『ファイル』→『新規作成』→『プロジェクト』
2. **Win32 プロジェクトを選択**して作成（いつもの Win32 コンソールアプリケーションではありません)
3. 空のプロジェクトにチェックを入れて完了
4. ソースファイル(.cpp)を追加
5. 「プロジェクト」→「プロパティ」と開く
6. 「構成プロパティ」→「全般」を選び、左上のリストを「すべての構成」に変更（アクティブ(Debug)になっていると思います)
7. 「文字セット」の項目を「**マルチ バイト文字セットを使用する**」に変更(右下「適応」を押して確定)
8. 「構成プロパティ」→「C/C＋＋」→「コード生成」
9. 左上のリストを「**Debug**」に変更
10. 『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(「適応」を押して確定)
11. 左上のリストを「**Release**」に変更
12. 『ランタイム ライブラリ』の項目を『**マルチスレッド(/MT)**』に変更
13. ＯＫを押して終了

これでようやく実際にコードを書いて実行できるようになります。
ちなみにこの操作を行わないと大量のエラーが出てきたりするので注意

④ 必須のコード

ＤＸライブラリを用いた最小限のコードは以下のようになります。（ウインドウが出てすぐ消えます）

```
#include "Dxlib.h"

int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR lpCmdLine,int nShowCmd)
{
	ChangeWindowMode( TRUE );		//ウインドウモードに変更(デフォルトはフルスクリーン)
	if( DxLib_Init() == -1 )		//ＤＸライブラリの初期化
	{
		return -1;			//初期化失敗したら終了
	}
	//ここに処理を記述
	DxLib_End();				//ＤＸライブラリ使用の終了
	return 0;
}
```

一部の関数(ChangeWindowMode など)を除いて DxLib_Init（初期化）前にはＤＸライブラリの関数は使えないので注意してください。

ChangeWindowMode 関数は作る際にフルスクリーンだとデバッグが不便なのでウインドウモードにするために入れてあります。

重要なのはライブラリの初期化→終了の過程が必要ということです。

以下のコードでは上記のコードはすでに書いてあるものとし、「//ここに処理を記述」と書いてある場所に記述する部分だけを書いていきます。

⑤ 実際にコードを書いてみる

1. とりあえず「Hello World! 」

```
	DrawFormatString( 0 , 0 , GetColor( 255,255,255 ) , "Hello World!" );
	WaitKey();
```

・DrawFomatString(表示位置(x , y)、色、表示文字)

→printf に相当する関数。ただし¥n(改行)はできない。自分でｙ座標を調節すること。

・ WaitKey()

→何かキーが押されるまで待つ関数。

・ GetColor (赤、緑、青)

→赤、緑、青にそれぞれ 0～255 の値を指定し、その色の値を返す。255,255,255 で白。

2. キー入力を受け取る

```
	while( ProcessMessage() == 0 )  //ループを作る際に必須
	{
		if( CheckHitKey( KEY_INPUT_ESCAPE ) == 1 ) //Escapeが押されたらループ終了
		{
			break;
		}
	}
```

・ **ProcessMessage()**

→Windows のウインドウメッセージ（終了など）を処理してくれる関数。無限ループ作成時に必須

while ループの際には入れておかないと右上の×ボタンなどが機能しなくなったり、一見終了したように見えてプロセスだけ残ったりするので入れること。

・ CheckHitKey(キーコード)

→キーコードで指定したキーが押されているか調べる（押されていたら 1、そうでなければ 0 が返る）

キーコードは KEY_INPUT_～の形で指定する。(KEY_INPUT_A など)

3. ダブルバッファリングについて＆簡単な図形を書いてみる

```
SetDrawScreen( DX_SCREEN_BACK );  //ダブルバッファリングの使用
while( ProcessMessage() == 0 && CheckHitKey( KEY_INPUT_ESCAPE) == 0)
{
        //円の描画
        DrawCircle( 160 , 240 , 100 , GetColor( 255 , 0 , 0 ) , TRUE );
        //四角の描画
        DrawBox( 380 , 140 ,580 , 340 , GetColor( 0 , 255 , 0 ) , TRUE );
        ScreenFlip();  //裏画面を表画面に反映
        ClearDrawScreen();  //裏画面のクリア
}
```

※Escape で終了します。

・ダブルバッファリング
ちらつき(コンソールで while ループ内に system(“cls”)を使った人はわかるはず)を軽減する手法。
表画面（表示用）と裏画面（描画用）の２つを用意し、裏画面に画像などを描いた後、それを表画面に反映することによってちらつきをなくす。詳しくは口頭で。

ＤＸライブラリでダブルバッファリングするには、
SetDrawScreen(DX_SCREEN_BACK)で裏画面に描画するように設定し、図形や画像を描画した ScreenFlip()を使って裏画面の内容を表画面に反映して行う。
反映したら裏画面を ClearDrawScreen()でクリアしておく。

・DrawCircle(座標ｘ、 ｙ、半径、色、塗りつぶすかどうか　)
・DrawBox (左上ｘ、 ｙ、右下ｘ、 ｙ、色、塗りつぶすかどうか)
→円、四角を描画する関数

4. 画像を表示する

```
int GrHandle;    //画像のハンドル(先に準備)
GrHandle = LoadGraph( "test.png" );       //画像の読み込み
SetDrawScreen( DX_SCREEN_BACK );
while( ProcessMessage() == 0 && CheckHitKey( KEY_INPUT_ESCAPE) == 0)
{
        DrawGraph( 0 , 0 , GrHandle , FALSE );
        ScreenFlip();
        ClearDrawScreen();
}
```

※Escape で終了します。

・ LoadGraph(ファイル名)
→画像をメモリに読み込み、int 型で画像にアクセスできるハンドルを返す。
・ DrawGraph(座標ｘ、 ｙ、LoadGraph で読み込んだ画像のハンドル、透過有無　)
→画像を描画する
DrawGraph には他にも同じような関数がある。
回転→DrawRotaGraph 拡縮→DrawExtendGraph 左右反転→DrawTurnGraph など

5. 音楽を再生する

```
int SoundHandle; //音楽のハンドル(先に準備)
SoundHandle = LoadSoundMem( "test.wav" ); //音楽ファイルの読み込み
PlaySoundMem( SoundHandle , DX_PLAYTYPE_BACK );
WaitKey();
```

・ LoadSoundMem(ファイル名)
→音楽ファイル（Wav , Mp3 , Ogg ）などを読み込む。使い方は LoadGraph と同じ。
・ PlaySoundMem(音楽ファイルのハンドル、再生方法)
→読み込んだ音楽ファイルを再生する。再生方法は DX_PLAYTYPE_BACK(バックグラウンド再生)、DX_PLAYTYPE_LOOP (ループ再生)など。

このほかにも再生を停止する StopSoundMem や再生されているか調べる CheckSoundMem などもあるので必要になったら調べてみること。

6. 乱数やウインドウタイトル（関数紹介のみ）
・SetMainWindowText(変更後のウインドウタイトル)
→システムバーの文字を変更する（デフォルトは DxLib ）

・ GetRand(乱数最大値)
→指定した最大値までの乱数を得る。種の設定は必要なし（明示的にする場合は SRand()を使う）

7. ＤＸアーカイブ
DX ライブラリには複数のファイルをまとめて一つのファイルにまとめるＤＸアーカイブという機能があります。
ただし、ＤＸアーカイブを使った場合、データの読み込みには標準関数の fopen などではなく FileOpen_～(ＤＸライブラリの関数)を使う必要があります。

使い方は DxaEncode.exe(DxLib_VC フォルダに入っています)を使って
dxa ファイル（ＤＸアーカイブファイル）を作成し、後はいつもどおりそのファイルをフォルダとして扱います。
詳しくはサイトの「ＤＸライブラリミニテクニック」を調べてください。

8. 番外編１：ｆｐｓについて

フレームレートは、単位時間あたり何度画面が更新されるかを表す指標である。
通常、1 秒あたりの数値で表し、fps(Frames Per Second)という単位で表す。
なお、コンピューターで使われるディスプレイにおいてはリフレッシュレートという表現が用いられる。
（by Wikipedia）

通常は１秒間に６０回のスピードで画面を切り替えるようにすることが多いです。
そうしないとコンピューターの性能によってゲームの難易度が変わってしまう(実行スピードに差がでる。)可能性があります。
秒間 60 回のループをさせるとき、１ループあたりの時間は 1/60 秒＝16.666..ミリ秒となります。

9. 番外編２：ＳＴＧ等におけるあたり判定

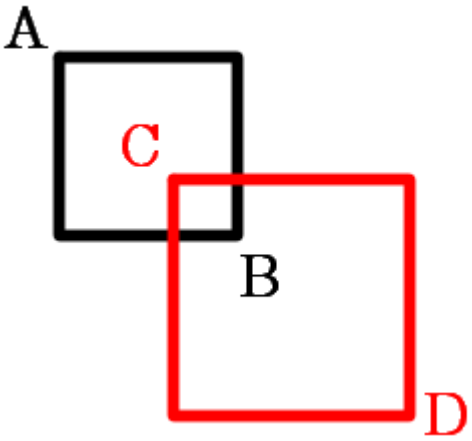

図１：矩形と矩形

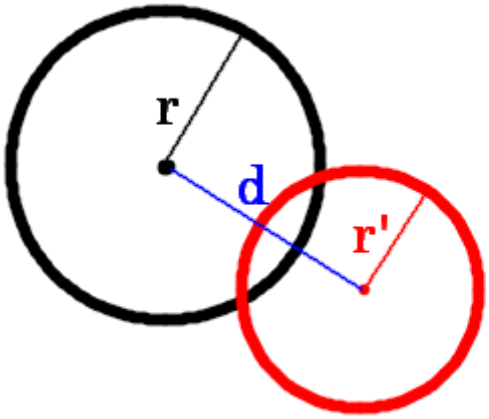

図２：円と円

・矩形と矩形（図１）
左の黒い四角の左上をＡ、右下をＢ、右の赤い四角の左上をＣ、右下をＤとすると、
ＡがＤより左上にあり、ＢがＣより右下にあれば「当たり」

・円と円（図２）
左の黒い円の半径をｒ、右の赤い円の半径をｒ´、二つの円の中心間の距離をｄとすると、
ｄ＜ｒ＋ｒ´ならば「当たり」