ソフトゼミ B　第 2 回 画像処理

■　はじめに

第二回ではゲームを作る上で基本となる、画像描写や文字描写を学んでいきます。
また、ゲームの基礎の基礎である**ゲームループ**を作り WaitKey で待つだけでない、しっかりとしたゲームの流れを作れるように学んでいきます。

■　画像ファイルの読み込みの準備

まずは、windows 上で画像ファイルをプログラムに読み込ませる位置に配置します。
自機(player.png)、敵(enemy.png)、サイドバー(sidebar.png)の 3 つの画像を入れたフォルダ「img」を作り、ソリューションファイル(.sln ファイル)があるフォルダに入れてください。
これで準備完了です。つぎはプログラムを変更していきます。

■　プログラムの解説

では、下のプログラムをゼミ B の main.cpp に書いてみてください。前回書いたものは消しても構いません。

```
#include "Dxlib.h"
/*  窓関連  */
//ウィンドウサイズ
#define WINDOW_WIDTH    640
#define WINDOW_HEIGHT 480
//サイドバーサイズ
#define SIDEBAR_WIDTH 160

/*  構造体の宣言  */
//読み込む画像の一覧
struct SImgList{
    int player;
    int enemy;
    (次ページへ続く)
```

```
	int sidebar;
};
//プレイヤー
struct SPlayer{
	int x, y;
	int life, maxLife;
	int score;
};
//敵
struct SEnemy{
	int x, y;
	int life, maxLife;
};
/*  グローバル変数の宣言  */
struct SImgList imgList;
struct SPlayer player;
struct SEnemy enemy;

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd ){

	// ウィンドウモードにする
	ChangeWindowMode( TRUE ) ;
	// 解像度とカラービット数を設定
	SetGraphMode( WINDOW_WIDTH, WINDOW_HEIGHT, 32 );
	// DX ライブラリの初期化に失敗すると、即終了
	if( DxLib_Init() == -1 ){ return -1; }
	//初期化
	player.x = 240;
	player.y = 240;
	player.life = player.maxLife;
	player.score = 0;

	enemy.x = 120;
```
(次ページへ続く)

```
	enemy.y = 50;
	enemy.maxLife = 5;
	enemy.life = enemy.maxLife;
//画像のロード
	imgList.player = LoadGraph( "img/player.png" );
	imgList.enemy   = LoadGraph( "img/enemy.png" );
	imgList.sidebar      = LoadGraph( "img/sidebar.png" );
//描画先の指定
	SetDrawScreen( DX_SCREEN_BACK ) ;

/************************  ゲームループ*****************************/
	while( ProcessMessage() == 0 && CheckHitKey(KEY_INPUT_ESCAPE) == 0){
		//画面に描かれたものを消去する
		ClearDrawScreen();
		//自機を描画 [[自機のライフが 0 になったときの処理は第 6 回で実装します。]]
		DrawGraph( player.x, player.y, imgList.player, TRUE );
		//敵を描画
		DrawGraph( enemy.x , enemy.y , imgList.enemy , TRUE );
		//サイドバーを描画
		DrawGraph(WINDOW_WIDTH - SIDEBAR_WIDTH,0,imgList.sidebar, FALSE );
		DrawString( WINDOW_WIDTH - SIDEBAR_WIDTH + 10, 50, "Life: ", GetColor(255,255,255) );
		DrawFormatString( WINDOW_WIDTH - SIDEBAR_WIDTH + 10, 75,GetColor(255,255,255), "%4d",
player.life );
		DrawString( WINDOW_WIDTH - SIDEBAR_WIDTH + 10, 125, "Score: ", GetColor(255,255,255) );
		DrawFormatString(WINDOW_WIDTH - SIDEBAR_WIDTH + 10,150,GetColor(255,255,255), "%6d",
player.score );

		//表画面に反映
		ScreenFlip();
		}

/************************ゲームループおわり*************************/
	//DX ライブラリを終了する
	DxLib_End();
```

(次ページへ続く)

```
    return 0;
}
```

とても長いですねぇ、これから解説していきます。

■　ゲームループとは

ゲームループとは「プレイヤーが動作する」→「動作を反映させる」→「画面に情報を提示する」→・・・の繰り返しを行うものです。

DrawGraph 関数や DrawString 関数、ScreenFlip 関数などを使って描画をしています。また、この描画の回数(枚数)の単位をフレームと言ったり、１秒間に 60 回描画されることを 60FPS (FPS = Frame Per Second)と言ったりします。したがって、60FPS のゲームの場合、ゲームループを 1 秒間に 60 周している事になるのです。

ゼミ B のゲームでは基本的に FPS を制御しません。そうなるとハードウェア依存になりますが、厳密な描写管理を求められるようなゲームではないので問題ないでしょう。

もしそれじゃ困るという人や、FPS をしっかり学んでおきたい人は
「龍神録プログラミングの館(43 章)」： http://dixq.net/rp/43.html
あたりを読んでみるといいかもしれません。

■　#define(マクロ)とは

プログラムの最初に

#define WINDOW_WIDTH　640

という命令があります。「define」は定義するという意味の英単語ですが、この場合は WINDOW_WIDTH を 640 と定義しています。

この命令を使うと、プログラム中のすべての「WINDOW_WIDTH」が「640」に置き換えられた状態でコンパイルが行われます。(より詳しくは第 5 回で学びます。) 文字が長くなってるじゃないか、非効率だと思われるかもしれませんが、この WINDOW_WIDTH はプログラム内で 6 回使われています。

例えば、ウインドウの幅を 640 から 780 にしたい。となった時にすべて書き換えるのは面倒で、バグの原因にもなりますしので定数を使うときは基本的に#define を使って定義してください。

また、#define のことをマクロと呼ぶことがあります。

■ 新出関数

➢ ゲームループに使った関数

ProcessMessage 関数

int ProcessMessage(void) ;

ウインドウが閉じられたときや、エラーが発生した時はプログラムを終了しなければなりません。この関数は windows からのそういったメッセージを受け取ります。この関数は、このような重要な処理を行うためゲームループするごとに 1 回程度の頻度で必ず呼び出す必要があります。

CheckHitKey 関数

int CheckHitKey(int KeyCode) ;

キーが押されているかどうかを調べます。引数でどのキーを調べるか指定します。戻り値が 1 なら押されていて、0 なら押されていません。引数のリストはリファレンスにありますのでここでは一部を抜粋します。

KEY_INPUT_ESCAPE //エスケープキー	KEY_INPUT_A //A キー
	KEY_INPUT_B //B キー
KEY_INPUT_UP //上キー	以下同様
KEY_INPUT_DOWN //下キー	
KEY_INPUT_RIGHT //右キー	KEY_INPUT_1 //1 キー
KEY_INPUT_LEFT //左キー	KEY_INPUT_2 //2 キー
	以下同様

今回は ProcessMessage 関数と CheckHitKey 関数を使ってゲームループのループ条件を
while(ProcessMessage() == 0 && CheckHitKey(KEY_INPUT_ESCAPE) == 0)
「プログラムが正常に動作していて、かつ ESCAPE キーが押されていない」にしています。
なお、KEY_INPUT_～は DxLib.h 内にある#define で定義されていて、実際には int 型の値になっています。

➢ 画像・描画に関係する関数

LoadGraph 関数

int LoadGraph(char *FileName) ;

LoadGraph 関数は引数内の文字列に従って画像を読み込み、識別番号を付けます。

この番号は(グラフィック)ハンドルと言い、int 型の数値です。取り込んだ画像を使うときにはこのハンドルを使います。今回は構造体 imgList のメンバである player にこの数を格納しています。

SetDrawScreen 関数

int SetDrawScreen(int DrawScreen) ;

この関数は描画する場所を表画面(そのまま描写)、裏画面(ユーザーからは見えない裏の画面)のどちらにするか設定します。今回は裏画面に描写します。表に描写すると描写途中の画像が見え、ちらついているように感じることがあり、これを防ぐため、裏画面に描写します。裏画面を指定する場合は、SetDrawScreen(DX_SCREEN_BACK);のように指定します。

ClearDrawScreen 関数

int ClearDrawScreen(void) ;

各種描画関数で描画したグラフィックをすべて消し画面を初期化します。

DrawGraph 関数

int DrawGraph(int x,int y,int handle,int flag) ;

というようになっていて、(x, y)を左上の起点にして handle(先ほどのグラフィックハンドル)で指定した画像を描写します。最後の引数は透明色を使うかのフラグで TRUE にすれば透明色が有効になります。

GetColor 関数

int GetColor(int Red , int Green , int Blue) ;

RGBの数字を入れることで、DrawString関数などで使う色コードを出力してくれます。出したい色のRGBがわからないという人はペイントツールなどのパレットで調べてください。

DrawString 関数

int DrawString(int x , int y , char *String , int Color)

画面内の座標(x,y)を左上として String に指定された文字列を描写します。文字の色は上の GetColor 関数で指定してください。

DrawFormatString 関数

int DrawFormatString(int x , int y , int Color ,char *FormatString , ...) ;

DrawString と同じですが、printf のように%d などで出力に変数を使えます。DrawString 関数と引数の順番が違うので注意してください。 上のプログラム中で「%4d」とありますが、これは整数を 4 ケタで出力するための記号(フォーマット指定子)です。(printf でも使います。) 4 ケタ未満の場合は半角スペースを詰めて 4 ケタになるように調整され、5 ケタ以上の時は普通に「%d」と指定した時と同様に扱われます。

ScreenFlip 関数

int ScreenFlip(void) ;

裏画面に描かれている画像を表画面に反映します。

- その他の重要な関数

次の関数はプログラムに出てきませんが、重要なので覚えておくといいかもしれません。。

SetFontSize 関数

int SetFontSize(int FontSize) ;

DrawString などの文字のサイズを変更します。引数は文字のサイズでおおよそドット単位で指定します。よくわからない人は、適当な値を入れて自分の好きな大きさに調整してください。

<使用例>

```
#include "DxLib.h"
#define WINDOW_WIDTH 640
#define WINDOW_HEIGHT 480

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd ){
    //初期化
    ChangeWindowMode( TRUE ) ;
    SetGraphMode( WINDOW_WIDTH,WINDOW_HEIGHT, 32 );
    if( DxLib_Init() == -1 ){ return -1; }
    //ゲームループ
    while(ProcessMessage() == 0 && CheckHitKey(KEY_INPUT_ESCAPE) == 0){
        ClearDrawScreen();        //画面を初期化

        SetFontSize(15);          //フォントのサイズを変更
    (次ページへ続く)
```

```
        DrawString(100,100, "15 のサイズで描写",GetColor(255,255,255)); //文字出力
        SetFontSize(30);            //フォントのサイズを変更
        DrawString(100,200, "30 のサイズで描写",GetColor(255,255,255)); //文字出力

        ScreenFlip();               //裏画面を表画面に反映
    }
    DxLib_End();
    return 0;
}
```

サイズを 15 にして「15 のサイズで描写」と出力する。
サイズを 30 にして「30 のサイズで描写」と出力する。
という処理をしています。

今日の講習は以上です。いきなり長いプログラムを取り扱いましたが、わからないことがあれば悩まずにどうぞ聞いてください。余力があるなら、値を変更したり命令を増やしたり減らしたりして遊んでみるといいかもしれません。きっと理解の助けになると思います。お疲れ様でした。