第２回　画像の表示

今回は、画面に画像を表示させます。ゲームに画像は必要不可欠ですね。シューティングを例に挙げると、自機・敵機・背景・弾（＋α）など一度に何種類もの画像を管理しなければなりません。しっかり覚えてね。

まず画像を用意する。当然ですね。事前に配布したサンプルの画像を使うか、勇気がある人は自分のマイピクチャの画像を使っても構いません。ただ、拡張子が.jpg のファイルは使えないので覚えておいて下さい。とりあえず今回はその画像ファイルに「test.bmp」という名前を付けて、『VC』というフォルダに入れます。

ＤＸライブラリで画像を表示させる手順は、以下の通りです。

①画像を読み込んでメモリに保存する
②メモリから画像を呼び出して表示する

他の方法もありますが、そこは補足を参照。

①画像を読み込んでメモリに保存する

先程、画像データを『VC』フォルダにぶち込みましたね。まず、『LoadGraph 関数』を用いて、メモリにそのデータをロードします。

```
LoadGraph( "画像のファイルのパス" )
```

これが LoadGraph 関数です。もちろん「ファイル名」の部分は LoadGraph 関数は引数としてファイル名を取り、戻り値に int 型のメモリに保存したグラフィックデータの識別番号を返します。よく分からなかったら「ファイル名書いたらその識別番号を教えてくれる奴だ」って思っていて下さい。識別番号は、そのデータを管理（使ったり削除したり）するのに必要です。なくさないように、int 型の変数にしまっておきましょう。もちろん変数はあらかじめ宣言しておいてね。

```
int 型の変数 = LoadGraph( "画像のファイルのパス" );
```

②メモリから画像を呼び出して表示する

DrawGraph 関数に表示させたい画像データの識別番号と表示させたい場所（座標）を教えてあげると、ちゃんと指定した位置に画像を表示してくれます。

```
DrawGraph( Ｘ座標の値 , Ｙ座標の値 , 識別番号 , FALSE );
```

識別番号は、さっき変数に入れましたね。だから、「識別番号」の部分にはさっきの変数名を書けばいいのです。

さて、ここで今までの復習として、『test.bmp』を画面の左端（0, 0）に表示させるプログラムを書いてみましょう。

前回の画面中央にドットを表示する方法は覚えていますか？初期化とか終了処理とかキー入力待ちとか……ついでに今回は使わないけどドットを描写する関数とか。折角なので前回のプログラムも載せておきますね。参考までに。意味深に太字とか使ってみたり。

```
#include "DxLib.h"

// プログラムは WinMain から始まります
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                        LPSTR lpCmdLine, int nCmdShow )
{
        int gazou;

            if( DxLib_Init() == -1 )            // DXライブラリ初期化処理
            {
                    return -1 ;                 // エラーが起きたら直ちに終了
            }

            DrawPixel( 320 , 240 , 0xffff ) ;   // 点を打つ

            WaitKey() ;                         // キー入力待ち

            DxLib_End() ;                       // DXライブラリ使用の終了処理

            return 0 ;                          // ソフトの終了
}
```

正解はこんな感じです。

```
#include "DxLib.h"

// プログラムは WinMain から始まります
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                        LPSTR lpCmdLine, int nCmdShow )
{
        if( DxLib_Init() == -1 )        // DXライブラリ初期化処理
        {
                return -1 ;             // エラーが起きたら直ちに終了
        }

        gazou = LoadGraph( "test.bmp" );

      DrawGraph( 0, 0, gazou, FALSE ) ;

        WaitKey() ;                     // キー入力待ち

        DxLib_End() ;                   // DXライブラリ使用の終了処理

        return 0 ;                      // ソフトの終了
}
```

（補足）

今回は画像データを一度メモリに保存してから使用しましたが、この「メモリに保存する」という作業をしないで直接使うことも可能です。この方法をとる事により、たった一行で画像を表示させる事が可能となります。その代わり、動作がかなり遅くなるそうです。あまりオススメしません。

```
LoadGraphScreen( 0 , 0 , "画像のパス" , FALSE ) ;
```

それでは、いよいよゲームのプログラムを制作していきましょう。まずは背景・自機・敵機の画像を画面に表示させてみましょう。３枚の絵を同時に表示させる事になりますが、やり方は全く変わりません。同じ作業

を画像の枚数分繰り返せばいいのです。

とりあえず一枚ずつ。まずは背景の画像を表示してみましょう。まず「img」というフォルダを作ります。以降、画像関係のデータはここに入れることにします。事前に配布した「HAIKEI.bmp」を、このフォルダに入れておきます。

せっかく書いたので、さっきのプログラムを流用しましょうか。

```
#include "DxLib.h"

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                        LPSTR lpCmdLine, int nCmdShow )
{
        if( DxLib_Init() == -1 )        // DXライブラリ初期化処理
        {
                return -1 ;             // エラーが起きたら直ちに終了
        }
        int haikei;

        haikei = LoadGraph( "HAIKEI.bmp" );

      DrawGraph( 0, 0, haikei, FALSE ) ;

        WaitKey() ;                     // キー入力待ち

        DxLib_End() ;                   // DXライブラリ使用の終了処理

        return 0 ;                      // ソフトの終了
}
```

int haikei に HAIKEI.bmp を代入。背景画像は画面いっぱいに表示させるので、DrawGraph 関数に指示する座標は(0, 0)です。

書けたらコンパイルしてみて下さい。ウィンドウいっぱいに背景画像が表示されるはずです。

同じように自機も表示させてみましょう。

◇裏画面

なんかカッコイイ名前ですね。

我々が見ている画面を「表画面」、我々には見えないところにある画面を「裏画面」と呼びます。今までの

プログラムでは、表画面に直接画像を貼り付けていました。今回のように静止画を一枚ペチっと貼るだけならこれでかまいません。しかし、ゲームなら何枚もの画像を同時に貼り付け、同時に動かす必要があります。ところが、プログラムは上に書かれている命令から順番に実行していくため、「同時に処理」というのは出来ないのです。シューティングにタイムラグがあったら困りますね。

では、同時に画像を出したい時にどうするか。ここで「裏画面」が活躍します。裏画面という舞台袖のような所で画面を作って、完成した画面だけを表画面に持ってきて表示させます。まずは裏画面に画像を貼ります。

```
SetDrawScreen( DX_SCREEN_BACK ) ;
```

この命令を書くと、描画先（画像を貼る画面）が表画面から裏画面に切り替わります。以下、今までと同じように画像を貼る命令をすると、勝手に裏画面に貼ってくれます。

```
SetDrawScreen (DX_SCREEN_BACK);                    // 描画先を裏画面へ

        int haikei;
        int ziki.img;

        haikei = LoadGraph( "HAIKEI.bmp" );        // 背景を描画
        ziki.img = LoadGraph( “ZIKI.bmp” );        // 自機を描画
```

背景と自機、２枚の絵を裏画面に描画するプログラムです。試しに実行してみると分かりますが、画面には何も出てきません。表画面には何も貼ってないので当然ですね。
では、裏画面にセットされたものを表画面に貼ります。

```
SetDrawScreen( DX_SCREEN_BACK ) ;
```

この関数を使うと、裏画面の内容をごっそり表画面に持っていくことが出来ます。プログラムのどの部分に書けばいいのか、考えてみて下さいね。ここまで出来たらコンパイルしてみて、動作を確認してみましょう。

この程度の簡単なプログラムなら、裏画面経由でも表画面に直接描画しても大差ないかと思います。ですが、次回のゼミから裏画面の利便性を実感できるでしょう。

これも次回から頻繁に使う関数です。

```
ClsDrawScreen() ;
```

画面に描かれている物を全消しします。次回じっくり学ぶとは思いますが、覚えていて損はないです。時間が余ったら試してみて下さいね。

お疲れ様でした。これで今回のゼミの内容は終わりです。次回は画像を動かします。ソフトゼミ B では回を追うごとにゲームが完成に近づいていくので、楽しみにしていて下さいね。一人でも多くの人が完成に辿り着き、ゲームを作り上げる喜びや自信、達成感を味わえる事を願っています。