ゲーム制作講座-補助資料　2007/09/12　　（09/26　Web 用にほんの少し修正）

みなさん、お疲れ様です。C 言語ちゃんと勉強していますか？C 言語講座最終日から２週間近く経っています。あれから、みなさんがどれくらいやっているのか・・・。ちなみに、私はプログラミングを少しずつではありますが、ちゃんとやっていますよ～。

そんなことはいいとして・・・、今回は DX ライブラリという、DirectX を初心者でも簡単に使えるようにするライブラリを使った、ゲーム制作法を教えますよ。C 言語勉強し忘れた人は、今日からでいいので独学で習得し、DX ライブラリを使ってゲームを作れるようになりましょう。（私の C 言語講座資料を Web 上にアップしました。そちらを勉強してからこれを見ましょうね。）

では、まず DX ライブラリをダウンロードしましょう。下のサイトで無料配布しています。

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

その中の「DX ライブラリのダウンロード」というところにあります。あと、「DX ライブラリの使い方」というページに各開発環境の DX ライブラリの設定方法が載っています。自分の開発環境と同じものを読んで、同じように設定してください。

設定が無事すんだという前提で、話を進めていきますよ。

基本

では、ソースの書き方ですが・・・、土台部分は覚えてください。下のソースが土台となる部分です。そこから、データの処理、入出力とかを書き足していけば、ゲームが作れるようになります。では、ソースを見てみましょう。

```
//最低限必要なものを書いたソース。ゲームを作る時に、このソースを土台にすれば簡単かもね。

#include"DxLib.h"                                //DXライブラリをインクルード
int WINAPI WinMain(HINSTANCE hInstance , HINSTANCE hPrevInstance , LPSTR lmCmdLine , int nShowCmd)
        //ここは覚えて（もしくは、コピー）
{
        ChangeWindowMode(TRUE);             //ウィンドウモードで起動
        SetWindowText("Test");              //ウィンドウの名前が「Test」
        if(DxLib_Init() == -1) return -1;//基本的にこの文の後に、DXLib.h内にある命令を使う。
        SetDrawScreen(DX_SCREEN_BACK);      //裏画面に画像などを出力する
        while(ProcessMessage() == 0)        //終了するまで、ここをループする。
        {
                //ここら辺に、ユーザ入力・データ処理を書く

                //ここら辺に、画像描画を書く

                ScreenFlip();               //表と裏を反転
                ClearDrawScreen();          //裏になった方を消去して、ループの最初に戻る
        }
        DxLib_End();                        //DXライブラリの終了
```

```
        return 0;
}
```

これが基本構造です。最初は同じように写して、使えばいいです。文章の内容が理解できてから、いじってみるといいでしょう。リファレンスを見れば、何をしている命令かわかると思います。

少し補足として話す必要があるのは、while(ProcessMessage() == 0)とその while 文の中の命令でしょうか？

ProcessMessage()はエラーや終了コマンド(たとえば、ウィンドウの閉じるボタン)が入力されると、0以外の数値になるものです。エラーや終了コマンドが押されない限り、ループを続けます。

あと、DX ライブラリには表と裏の２つの描画スペースがあります。表はディスプレイ上にでているスペース、裏は見ることができないスペースです。そこで、描画するところを裏側にして、すべての描画処理が完了したところで表に出すようにすることで、画面のちらつきを防ぐことができます。そして、今まで表だったスペースをクリアして、次の描画を開始するのです。

SetDrawScreen(DX_SCREEN_BACK);	描画するスペースを裏側になるよう指定（while 文の前）
ScreenFlip();	**表と裏の画像を入れ替える（while 文ラスト）**
ClearDrawScreen();	**描画するスペース（今回は裏）内の画像をすべて消去（while 文ラスト）**

そして、この while 文の中身は１フレームの処理になっていて、「データ処理・ユーザ入出力⇒画像描画⇒画像反転・消去」という順序で１フレームが終了します。

ボタン入力

CheckHitKey（KEY_INPUT_○○）というものを使います。○○はキーボードのキーを書いてください。Z キーが押されたら反応するようにしたければ、「KEY_INPUT_Z」というように書きます。もし、Z キーが押されている場合は１を、押されていない場合は０という値を返します。

この関数を使えば、キー入力をしたときのみに処理するプログラムが作れます。If(CheckHitKey(KEY_INPUT_Z) = = 1)と書くと、Z キーが押されたときの処理が書けます。

```
if(CheckHitKey(KEY_INPUT_UP) = = 1)          //カーソルキーの上
{
        player.y －= 2 ;                     //上に２ピクセル分移動します
}
if(CheckHitKey(KEY_INPUT_DOWN) = = 1)        //カーソルキーの下
{
        player.y += 2 ;                      //下に２ピクセル分移動します。
}
```

これで、とりあえずユーザの入出力によって、データを書き換えることができるようになりました。あとは出力です。まあ、画像の出力の前に文字を出力できるようにしましょう。画像を出すことより文字の方が簡単です。デバッグをするときには文字を出したほうが良いでしょう。

文字列出力

DrawFormatString（int x , int y , int color , “文字列”）というものを使います。printf 関数と使い方は多少違いますが、慣れればすぐ使えるようになるでしょう。

１、２番目の引数 int x と int y は文字列を出力する位置です。コンソールプログラムでは位置とかは自動でやってくれましたが、今回は指定できます。（まあ、ウィンドウ上に表示するのに位置を自動で決められたら、大変ですね）その座標となる値・変数をここに書きます。

ちなみに、座標ですが数学のグラフでは、y が下から上に上っていくにつれて数値が大きくなりますが、今回は y が下に行くにつれて数値が大きくなります。それは間違えないようにしてください。

次に３番目の引数 int color ですが、これは表示する文字列の色を決めます。色の取得は GetColor (int red , int green , int bleu) を使います。各色の値を入力することで、色を指定します。０～２５５の範囲で入力してください。この関数の処理の結果は int 型でててきますので、

int color ;

color = GetColor(255 , 255 , 255) ;

というように事前に打ち込めば、白の文字を出力するときに color を使うだけで白文字を出力できます。

４番目は文字列です。これは printf と大差ないです。変数の値を出したいときは「型フィード文字」（%d とか）を使えば OK です。ただ、エスケープシーケンス（¥n とか）には対応していません。改行するときは、再度この関数を書いて y の値を変えてください。

演習①　とりあえず、カーソルキー（上下左右）を押すとプレイヤーの座標が移動するようにしてみてください。もちろん、x と y の両方の値を文字で出力して、わかるように。

画像出力

とうとう画像の出力の方法について、解説していきます。これさえわかれば、もうゲームは作れそうですね。では、説明します。

```
#include"DxLib.h"                                //DXライブラリをインクルード
int WINAPI WinMain(HINSTANCE hInstance , HINSTANCE hPrevInstance , LPSTR lmCmdLine , int nShowCmd)
        //ここは覚えて（もしくは、コピー）
{
        ChangeWindowMode(TRUE);                  //ウィンドウモードで起動
        SetWindowText("Test");                   //ウィンドウの名前が「Test」
        if(DxLib_Init() == -1) return -1;        //エラーがないか確認
        SetDrawScreen(DX_SCREEN_BACK);           //裏画面に画像などを出力する
        int graph = LoadGraph("ファイル名");
        while(ProcessMessage() == 0)             //終了するまで、ここをループする。
        {
                //ユーザ入力・データ処理を書く場所

                DrawRotaGraph(100 , 100 , 1 , 0 , graph , FALSE);

                ScreenFlip();            //表と裏を反転
                ClearDrawScreen();       //裏になった方を消去して、ループの最初に戻る
        }
        DxLib_End();                     //DXライブラリの終了
        return 0;
}
```

まず、画像出力では２つ関数を使います。「LoadGraph」と「DrawRotaGraph」です。
LoadGraph（”ファイル名”）は画像読み込みです。ハードディスクからメモリに画像データを読み込んでいます。そして、関数の結果を int 型の変数に代入しています。以後その変数を書くだけで、その画像を使用できます。

このやり方・・・、C 言語にも出てきたような・・・。

C 言語：fp = fopen(“ファイル名” , “オープンコード”)
Dxlib： graph = Load(“ファイル名”)

そう、ファイルの入出力とやり方は似ています。事前にファイルを開いて、使用するときは代入された変数を使うという方法はほとんど同じです。

あと、DrawRotaGraph（int x , int y , double extrate , double angle , int handle , int transflag）についてです。
１、２番目の引数は座標です。ただ、画像の中心をどこにするかの座標です。
３番目の引数は拡大・縮小の倍率です。１で通常の大きさになります。
４番目の引数は回転する角度です。単位はラジアン（360 度⇒2πで、０が通常になってます。
５番目の引数は画像のハンドル、つまり LoadGraph の値を代入した変数を書きます。
最後の引数は透明色を使用するかを書きます。TRUE なら使用、FALSE なら全部描画です。

これで、最低限の画像表示はできますが、透明色を知らないと後々つらいと思うので、その辺の話も・・・。

透明色
透明色を知らないで、画像を描画すると、各素材の背景として考えられていた部分まで、出力されてしまいます。それを直すために透明色を設定して、画像をメモリに読み込むときに透明色として認識させる方法があります。
SetTransColor（int red , int green , int blue）というものを使います。LoadGraph の前に使用してください。ここで、設定した色が、LoadGraph で読み込まれた画像に含まれていた場合、その色の部分を透明色として読み込みます。

たとえば・・・、もし、画像（”data1.bmp”）の背景色が緑（0,255,0）なら・・・、

```
SetTransColor(0,255,0) ;
int graph = LoadGraph(“data1.bmp”) ;
                    :
                    :
DrawRotaGraph（100 , 100 , 1 , 0 , graph , TRUE）;
```

と書いていくと、その画像の緑の部分が透明に表示されます。

演習②
では、①の画像表示バージョンを作りましょう。画像は適当に・・・。

まあ、あとはライブラリのリファレンスを見ながらやっていけば、大丈夫でしょう。正直即席で作ったため、書かれていないことが山ほどあります。なので、リファレンスを見て、いろいろいじってみましょう。では、これにて補助資料は終了します。お疲れ様です。（メイン資料は残念ながらありません）