ソフトゼミＢ　第１回 VisualC++・DX ライブラリの導入

■　はじめに

ゼミ B では、ゼミ A で学んだ C 言語の基礎を土台に、Microsoft Visual C++ 2010 Express Edition(以下、VisualC++, VC++)の使い方と、「DX ライブラリ」というゲームの作成を強力にサポートしてくれるプログラム群の使い方を学んでゆきます。なお、VC++と DX ライブラリを使うため、扱う言語は C 言語ではなく C++となるのですが、DX ライブラリを使用する分には C 言語の知識のみで使用することが可能(DX ライブラリの内部的には C++で動いている)なので、ゼミ B では C 言語的な書き方で書いてゆきます。ゼミ B の 2～7 回では簡単なシューティングゲームの作成を、8～10 回では簡単なアクションゲームの制作を行います。

DX ライブラリを用いることによって、DX ライブラリで用意されている関数を使うことができるのですが、各回の配布資料の末尾にはその回で新たに出てきた DX ライブラリの関数の説明を記しました。サンプルプログラムを読んで「ん！？」と思った場合には、資料の最後の方を見てみるといいかもしれません。それでも解決しない場合にはぜひ 2 年以上のソフト班員までどうぞ。

■　注意(最初は読み飛ばしても構いません)

ゼミ B では、プログラムをわかりやすくするために「**グローバル(大域)変数**」という、どの関数からでも使用可能な変数を用いていますが、本来はゼミ A 第 7 回で説明した**ポインタ**を用いたり、C++から追加された**クラス**とよばれる機構を用いたりして変数の有効範囲を適切に保つべきです。もちろん、オブジェクト指向のプログラミングに慣れているなんて場合は、使えるに越したことはないので、クラスでも何でもガリガリ使って構いません。あくまでも、このゼミ B で出てくるサンプルプログラムはゼミ A 第 6 回「関数」までの知識でも書けるようにした「一例」であることをご承知おき願いたいと思います。なお、C++に関する内容は、夏休み中に予定しているゼミ C(任意参加)で扱う予定です。

■　Microsoft Visual C++ 2010 Express Edition の導入

ゼミ A の最終回でも VC++のインストール手順は示しましたが、もう一度載せておきます。

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

の下の方、「Microsoft **VisualC++** 2010」の欄にある「Web インストール」をクリックしてインストーラをダウンロードしてください。ダウンロード後、インストーラの指示に従って、インストールを開始してください。

ところで、**製品の登録**はやってきましたか？まだの場合は、早めに登録してください。(要インターネット接続) 製品登録がまだの状態で VisualC++を使い続けると、突然起動できなくなることがあるので、インターネット接続があるうちに製品を登録しておくことを強く推奨します。(登録方法はゼミ A 最終回の資料をご覧になるか、検索エンジンで検索してください。)

■ DX ライブラリのダウンロード

こちらもダウンロードまでの手順はゼミ A 最終回に掲載しましたが、再掲。

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

から「**VisualC++用**」をダウンロード→解凍してください。解凍して出てきたフォルダは C ドライブ直下などのわかりやすいところに入れておくことをおすすめします。

■ プロジェクトの作成

VC++で「プロジェクト」なるものを作成します。このプロジェクトは、以降 VC++で一つのソフトウェアを作るたびに作成します。今回は、以下の手順で作成します。

1. VC++を起動する。デフォルトであれば、「スタート」→「すべてのプログラム」→「Microsoft Visual Studio 2010 Express」→「Microsoft Visual Studio 2010 Express」で起動できる。
2. VC++のウィンドウ上部、メニューバーの「ファイル(F)」から「新規作成(N)」→「プロジェクト(P)」を選択。
3. 出てきたウィンドウ左側「インストールされたテンプレート」で、「Visual C++」内「Win32」を選択、右側で「**Win32 プロジェクト**」を選択。ウィンドウ下部の「名前(N)」の欄に作るプロジェクトの名前を入力(例: zemi_b)。**右側「ソリューションのディレクトリを作成(D)」のチェックを外して、**OK を押す。
4. 「Win32 アプリケーション ウィザード」なるウィンドウが出てくるので、「次へ>」を押す。
5. 「アプリケーションの種類」が「Windows アプリケーション(W)」になっていることを確認して、「追加のオプション」で「**空のプロジェクト(E)**」にチェックを入れる。その後、完了を押す。

■ プログラムファイル(main.cpp)の追加

これから実際にプログラムを書いていくファイル「main.cpp」を作成します。「はじめに」で述べたとおり、C 言語的な書き方をしていきますが、DX ライブラリの内部的には C++が使用されていますので、拡張子は C++を示す「.cpp」となります。では、以下の手順で「main.cpp」を追加してみましょう。

1. VC++のウィンドウ上部、メニューバーの「プロジェクト(P)」から「新しい項目の追加(W)」を選択。
2. 出てきたウィンドウ左側「インストールされたテンプレート」で、「Visual C++」内「コード」を選択。右側は「C++ファイル(.cpp)」を選択。ウィンドウ下部「名前(N)」には「main.cpp」と入力し、「追加(A)」を押す。

■ DX ライブラリの設定

このプロジェクトから DX ライブラリを使用するための設定を行います。DX ライブラリを使うプロジェクトでは、**最初のプログラムファイル**(ここでは main.cpp)**作成後**に必ず行なってください。(main.cpp 作成前だと出てこない項目があります。)

やや長いですが、プロジェクト作成時(と、他の人からプログラムを受け取った時)のみなので頑張りましょう。

➢ 「プロパティ ページ」の出し方

まず、このプロジェクトに関わる設定を変更できる画面「プロパティ ページ」を出します。VC++のウィンドウ上部、メニューバーの「**プロジェクト(P)**」から「**(プロジェクト名)のプロパティ(P)**」を選択します。(例えば、プロジェクト名が「zemi_b」なら「zemi_b のプロパティ(P)」となります。) これでプロパティ ページが出てきます。

➢ 「プロパティ ページ」が出たらやること

画面左側のリストから「**構成プロパティ**」を選んでください。

➢ すべての構成の設定

開発中にプログラムを走らせるときは「Debug」とよばれる構成の設定を、実際に完成したプログラムを走らせる時には「Release」とよばれる構成の設定を利用するようになっているのですが、これから、それらの設定を行なっていきたいと思います。まずは、「Debug」と「Release」に共通する設定を設定していきます。主に、DX ライブラリがどこにあるかを VC++に伝える作業です。

1. 画面左上の「構成(C)」を「**すべての構成**」に切り替えます。

2. 画面左側のリストから「**構成プロパティ**」→「**全般**」を選んでください。
3. 画面右側「**文字セット**」の項目を「Unicode 文字セットを使用する」から「**マルチ バイト文字セットを使用する**」に変更します。
4. ダイアログ右下にある「**適用**」　ボタンを押します。
5. 画面左側のリストから「**構成プロパティ**」→「C/C++」→「**全般**」を選んでください。
6. 「**追加のインクルードディレクトリ**」の項目に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**』フォルダへのパスを入力してから、再度ダイアログ右下にある「**適用**」ボタンを押します。
   （ パスの例→　　C:¥DxLib_VC¥プロジェクトに追加すべきファイル_VC用）
7. 画面左側のリストから「**構成プロパティ**」→「**リンカー**」→「**全般**」を選んでください。
8. 「**追加のライブラリディレクトリ**」に先ほどと同じフォルダへのパスを入力してください。終わったらやっぱり「**適用**」ボタンを押します。

- 構成「Release」の変更

1. 構成(C)を「Release」に変更します。
2. 画面左側のリストから「**構成プロパティ**」→「C/C++」→「**コード生成**」を選んでください。
3. 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(/MT)**」に変更して「**適用**」。

- 構成「Debug」の変更

1. 構成(C)を「Debug」に変更します。
2. 『**ランタイム ライブラリ**』の項目を、今度は『**マルチスレッド デバッグ(/MTd)**』に変更して「**適用**」。

以上で設定は終了です。「**OK**」を押すと、「プロパティ ページ」が閉じます。

DX ライブラリへのパスが異なる他の人からプログラムを受け取った時は「**すべての構成**」で DX ライブラリのパスを上で述べた 2 箇所で変更しなければなりませんので注意してください。

これでプログラムを書く準備ができました。いよいよです。

■　プログラムを書く

```
#include "Dxlib.h"

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( 640, 480, 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; }

  //白い 32[px] * 32[px]の正方形を描画
  DrawBox( 0, 0, 32, 32, 0xffffff, TRUE );

  //キーが押されるまで待機
  WaitKey();

  //DX ライブラリを終了する
  DxLib_End();

  return 0;
}
```

main.cpp で上記の内容を入力し、保存してください。途中で出てくる各関数の説明は資料の最後の方で。

■　コンパイル(ビルド)・実行

書いたコードをコンパイルします。画面上部の緑の三角印(再生ボタン?)を押すか、F5 キーでコンパイルをすることができます。(実際には「ビルド」と呼ばれる作業です。詳しくは第 5 回で取り扱います。)

コンパイル(ビルド)が終了すると、自動でプログラムが実行されます。真っ黒な画面の左上隅に白い正方形が出れば成功です。もし、「ビルド エラー」が出た場合には、もう一度プログラムを見なおしてください。VC++では、文法上のミスなどを赤い下線で示してくれるので、確認してみましょう。

■ サンプルプログラムの大雑把な説明

- #include "Dxlib.h"

DX ライブラリを取り込みます。ちょうど、#include<stdio.h>で printf や scanf が使えるようになったのと同じで、DX ライブラリの関数を使えるようにするためには#include "Dxlib.h"と書く必要があります。なぜ 2 重引用符(“～”)で囲まれているか、など#include の詳しい意味はゼミ B 第 5 回で扱います。

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

すごーく長いですが、これはちょうどゼミ A の C 言語で int main(void){と書いていたようなもので、最初に実行される関数です。DX ライブラリを使う場合含め、Windows アプリケーションではじめに実行されるのがこの WinMain 関数です。引数については、今は特に気にする必要はありません。(ゼミ B では扱いません)

- その他、出てくる関数

「新出関数」で扱います。

■ 新出関数

このコーナーでは、新たに出てきた DX ライブラリの関数を紹介します。DX ライブラリの関数の一覧は、DX ライブラリが入っているフォルダ内の「help」フォルダ内「dxfunc.html」「dxfunc2.html」から参照できます。

- 使用必須関数

以下に示す「DxLib_Init」関数と「DxLib_End」関数は DX ライブラリを使うためには**必ず呼ばなければなりません**。

DxLib_Init 関数

int DxLib_Init(void) ;

DX ライブラリを初期化します。DX ライブラリを使って何かをする場合にはほぼ必ず呼ばなければなりません。なお、何らかの事情があって初期化に失敗した時、この関数は-1

という値を返します。(平常時は 0 を返します。) なので、この関数を呼ぶ時には、

```
if( DxLib_Init() == -1 ){ return -1; }
```

という呼び方をします。これによって、異常があった時に WinMain 関数をその場で終了することができます。DxLib_Init に限らず、この類の関数は、失敗時に「-1」を返すことが多いです。それにならって、WinMain 関数も-1 を返すようにしています。

DxLib_End 関数

int DxLib_End(void) ;

DX ライブラリの使用を終了します。こちらも失敗時は-1 を返すのですが、この関数を呼ぶのは、プログラムの終了時なので DxLib_Init のように条件分岐せずに使うのが一般的です。後述するように、DX ライブラリはウィンドウを司っていますので、この関数を呼んだ後は、速やかにプログラムを終了させるようにしてください。

- ウィンドウ関連

DX ライブラリの関数は、基本的には DxLib_Init してからでないと使えないのですが、以下のウィンドウ絡みの関数は、使用することができます。

ChangeWindowMode 関数

int ChangeWindowMode(int Flag) ;

Flag が「TRUE」の時、ウィンドウでプログラムが開きます。「FALSE」の場合はフルスクリーンでプログラムが開きます。なんで int 型なのに「TRUE」「FALSE」という英単語なのかというお話ですが、それぞれある int 型の値を示すことになっています。詳しくは、次回#define を学ぶとわかるようになるかと思います。

SetGraphMode 関数

int SetGraphMode(int SizeX , int SizeY , int ColorBitNum) ;

画面の解像度を指定します。横の解像度が SizeX[px]、縦の解像度が SizeY[px]となっています。ColorBitNum のところは、カラービット数を指定します。DX ライブラリのデフォルトではピクセル単位の色の情報は 16 ビットで示され、($2^{16} = 65536$ 色)の色が扱えます。今回は、32 ビットを指定したので($2^{32} \fallingdotseq$ 1667 万色)の色が扱えます。

解像度とカラービット数については、黒板で詳しく説明します。

- その他

DrawBox 関数

int DrawBox(int x1 , int y1 , int x2 , int y2 , int Color , int FillFlag) ;

画面に(x1, y1)を左上の頂点、(x2-1, y2-1)を右下の頂点とする長方形を描画します。なぜ

右下の座標は指定した値から 1 少ないのか、ということは黒板で説明します。

Color には色を数値で指定します。ここでは 0xffffff としていますが、これは 16 進数で FFFFFF となる数を指定しています。この場合、光の 3 原色の混ぜ具合で色を表現するのですが、頭 2 ケタが「赤」の強さ(00～FF)、中 2 ケタが「緑」の強さ(00～FF)、後ろ 2 ケタが「青」の強さです。なお、次回「GetColor」関数を学ぶともう少し簡単に指定できるようになります。

WaitKey 関数

int WaitKey(void) ;

キーボードから何か入力される、もしくはマウスで窓のどこかをクリックされるまで待ちます。ただそれだけ。

いかがでしょうか？覚える内容が多すぎる、と感じましたか？これらの関数は一気に覚えるようなものではなく、使いながら馴染んでいくものだと思います。

次回からはいよいよシューティングゲームの制作が始まります。わからないことがあればいつでも質問に来てください。

今日の分はこれで終了です。お疲れ様でした。