第 13 章

プログラマー歴実質 1 年の僕が Windows と C++ でプログラミングをしたかった話

75 回生 RLtdExp

Windows API は、Windows の OS の各機能を、アプリから使えるようにしたものです。これにより、皆さんが普段目にするアプリケーションが開発されています。

現在は、.NET Framework という環境が整っていますが、.NET Framework も Windows API を使って作られており、.NET Framework でサポートされていない機能を使うには、Windows API を知らなければなりません。

しかし、16 ビット版からの度重なるアップデートにより、互換性を失ったプログラムが動作しなくなるなどのことがあったほか、非常に複雑になってしまい、覚えにくくなっています。私も挫折した一人です。私が使っている C++ でのプログラムは以下のようになります。

List 13.1: hoge

```
#include <windows.h>
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd){
        MSG msg;
        ＜処理＞
        return msg.wparam;
}
```

まずここで、「HINSTANCE とか LPSTR ってなんだよ」となりました。プログラミングの経験がある人は、int や char 、long といった型を知っているかと思いますが、 HINSTANCE や LPSTR は Windows API 独自のものです。

私が初めにつまずいたのが、これにかかわるものでした。ウィンドウを作る関数（決まった処理をするコード）の CreateWindow() の説明に、引数（関数に渡すデータ）の説明として

- ウィンドウクラスの名前
- ウィンドウの名前
- ウィンドウのスタイル
- 初期 X 座標
- 初期 Y 座標

…となっていたので、第一引数に、C++ 言語に基づいて "Hoge"（ char 型の配列) と書いたところ、「 char* 型のデータを LPCTSTR 型に変換することはできません」といわれ、動作しませんでした。

キャスト（型を強制的に他の型に変える）でも駄目でした。一つ目の挫折です。ネット上からサンプルプログラムを探したところ、みなさん TEXT("Hoge") と書いていたり、 _T "Hoge" と書いていたりしました。

どうやら、 LPCTSTR 型というのは、使用文字コードによって違うようで、 TEXT() は、 #define TEXT(S) _T S もしくは、#define TEXT(S) S と定義されており、使用されている文字列の方にキャストしてくれるマクロだったようです。

おそらく、互換性の観点からこれを使うのがよさそうです。

次に、メッセージ処理についてです。

MSG 構造体は、メッセージを受け取る型です。Windows プログラムの WinMain 関数は、基本的に MSG 構造体の wparam メンバーを返さなければなりません。この値は、めったなことがない限り 0（異常なし）です。

……と、かなり難しかったため、拡張ライブラリの「DX ライブラリ」を使い、開発を始めました。が [Windows SDK ○○が

存在しません] とエラーが出てしまいました。原因は、私の使っている Visual Studio 2017 に、DX ライブラリで使用している Windows SDK が入っていなかったためでした。

SDK をインストールして、さあ起動！ しかし

```
[○△□がありません]
[○△□がありません]
[○△□がありません][エラーが多すぎます。これ以上は表示できません]
```

嘘やん…何で…

もう一度サイトを確認すると…「Windows .exe アプリケーション」のところを変更してください」と書いていました。

どうやら、アプリケーションのタイプを間違えており、ライブラリが不足していたようです。というわけで、DX ライブラリでゲーム開発をやってみることにしました。

ゲームを作っていくうえで学んだことを述べていこうと思います。まず、数学についてです。

主に、関数とグラフのところを理解している必要があります。円運動や当たり判定をさせるときに三角関数があると便利です。

C 言語系統では、「math.h」という拡張ファイル (ヘッダーファイルといいます) に、いろいろな数学関数が入っています。例えば、正弦を求める「sin(double θ)」や、その逆関数の「asin(double θ)」などです。

これらの数学関数を駆使して、キャラクターなどの動きを制御します。

次に、最適化についてです。

どんなすごいプログラムを作っても、動かなければ意味がありません。できるだけ無駄を省き、より性能の低いパソコンでも使えるようにしなければなりません。

そのためには、どの部分が無駄か、どの部分が必要かを見極めなければなりません。例えば、全く使っていない変数や関数などです。これらは、メモリを無駄に食いつぶし、動作を遅くする原因となります。そのため、コードから削除することで、動作を軽くできます。

また、どうしても重いときは、余計なエフェクトを抜く、または、高性能パソコンでのみエフェクトをつけるようにすることも必要です。

13.1　あとがき

今回、Windows プログラムを作る上で、たくさん挫折を体験しました。また、いろいろなことを学ぶこともできました。

皆さんもパソコン研究部の展示や部誌を見て、多くの方にプログラミングについて知ってもらえるとありがたいです。