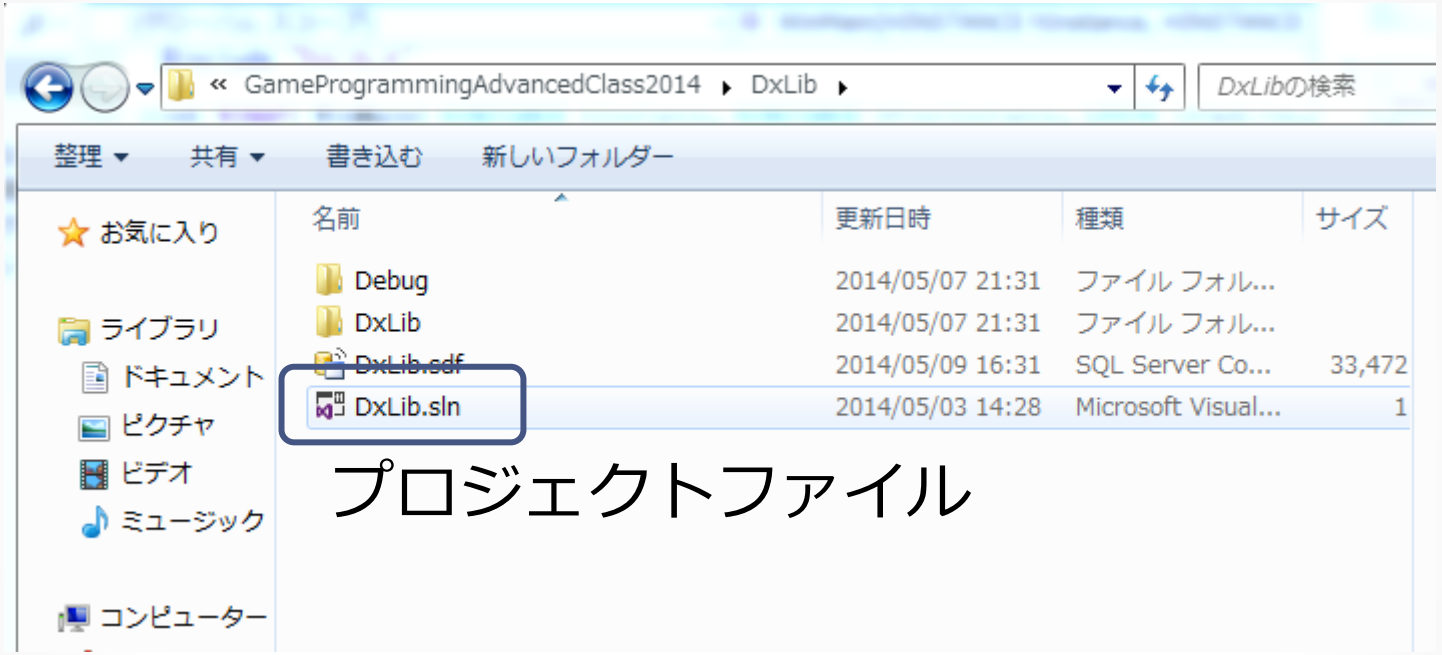
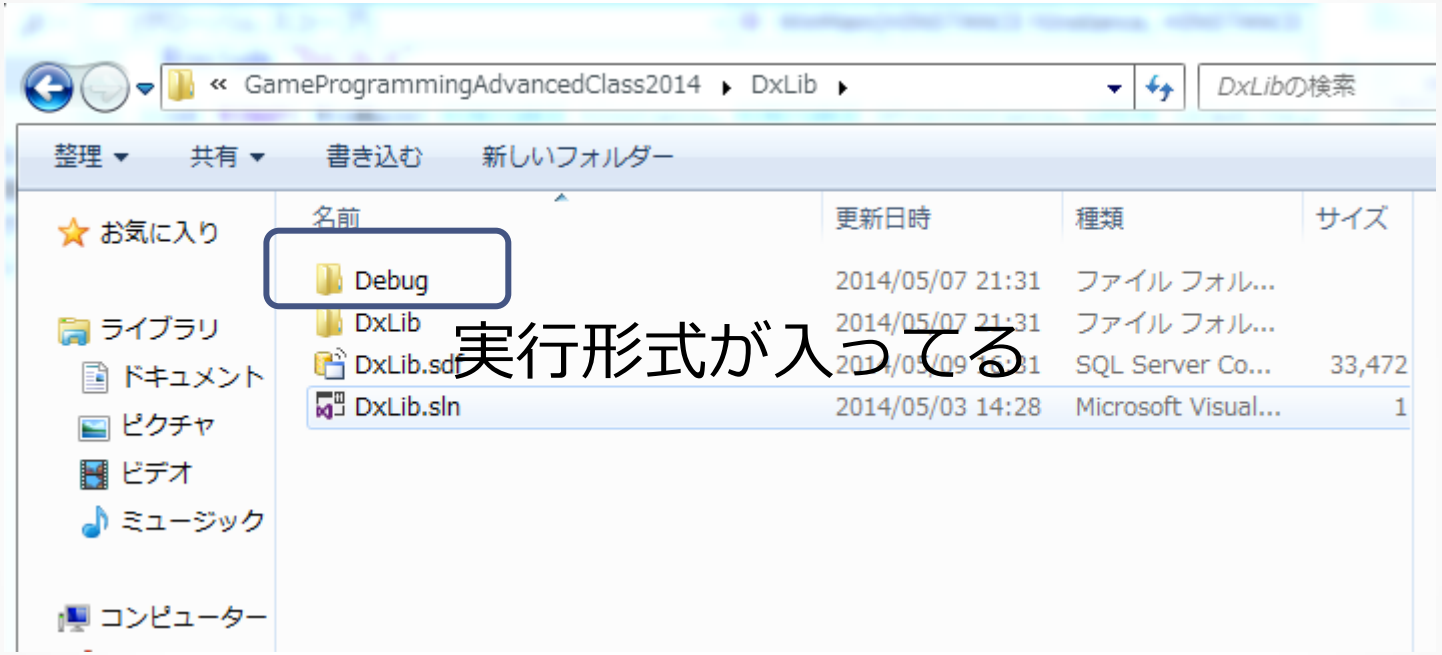
C++ゲームプログラミング講習
上級 第2回

画像描画

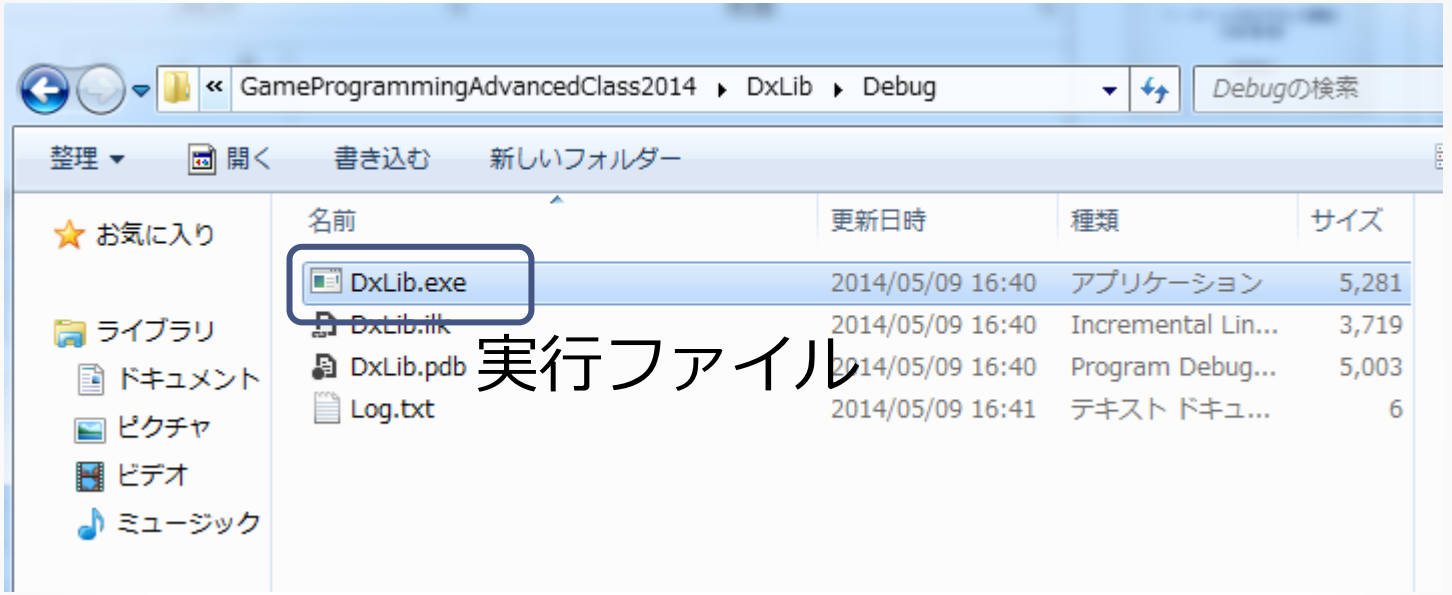
DxLibのフォルダ構成



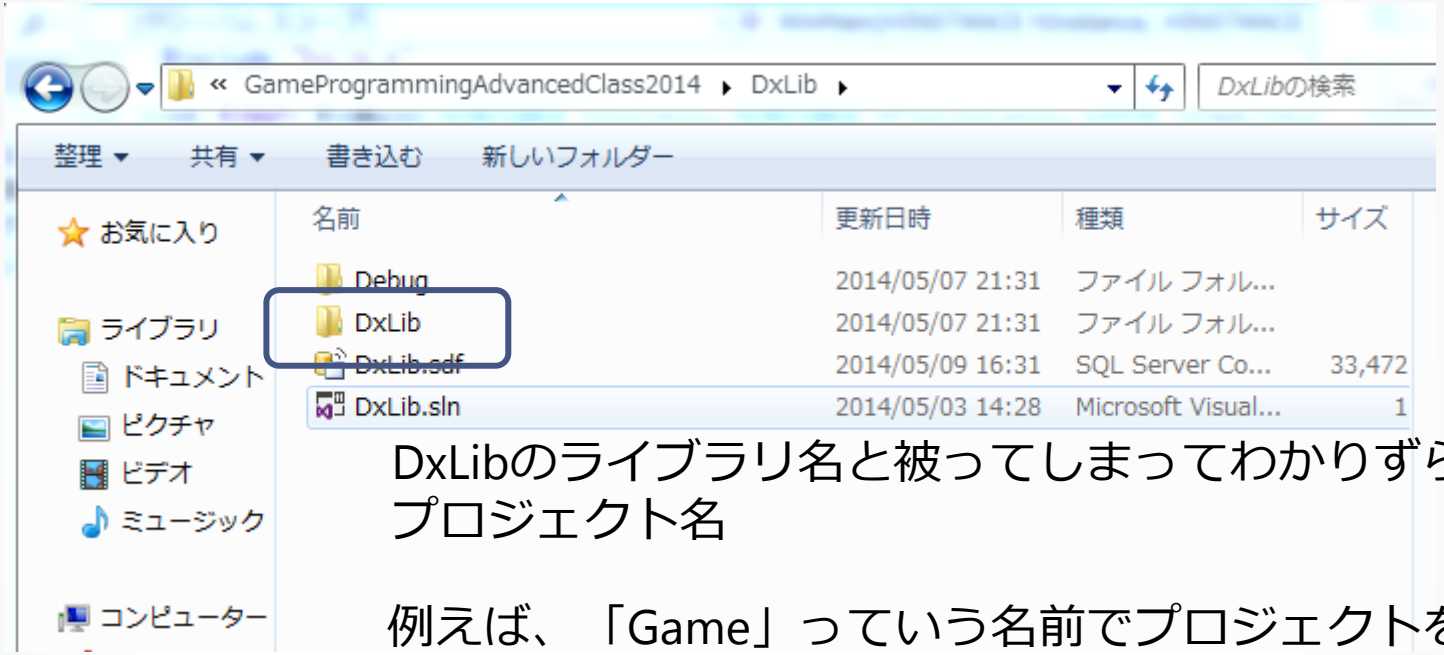
DxLibのフォルダ構成



DxLibのフォルダ構成



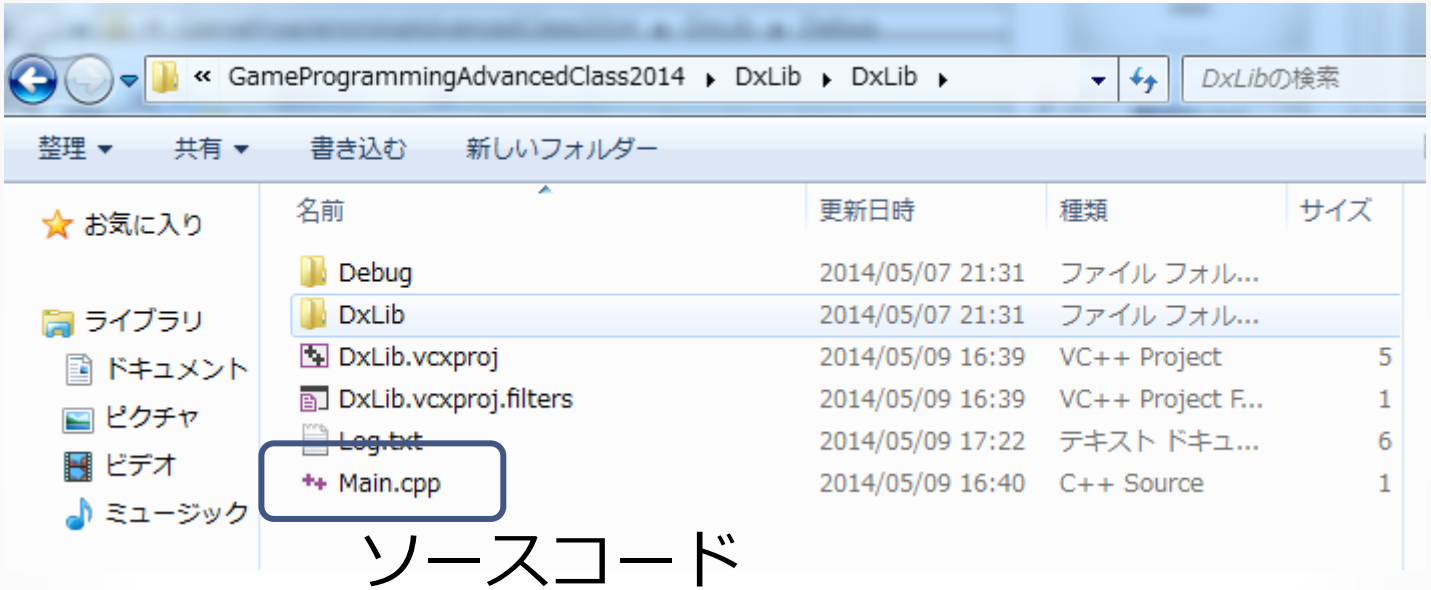
DxLibのフォルダ構成



DxLibのライブラリ名と被ってしまってわかりずらいが、プロジェクト名

例えば、「Game」っていう名前でプロジェクトを作ったらこれがGameになる

DxLibのフォルダ構成



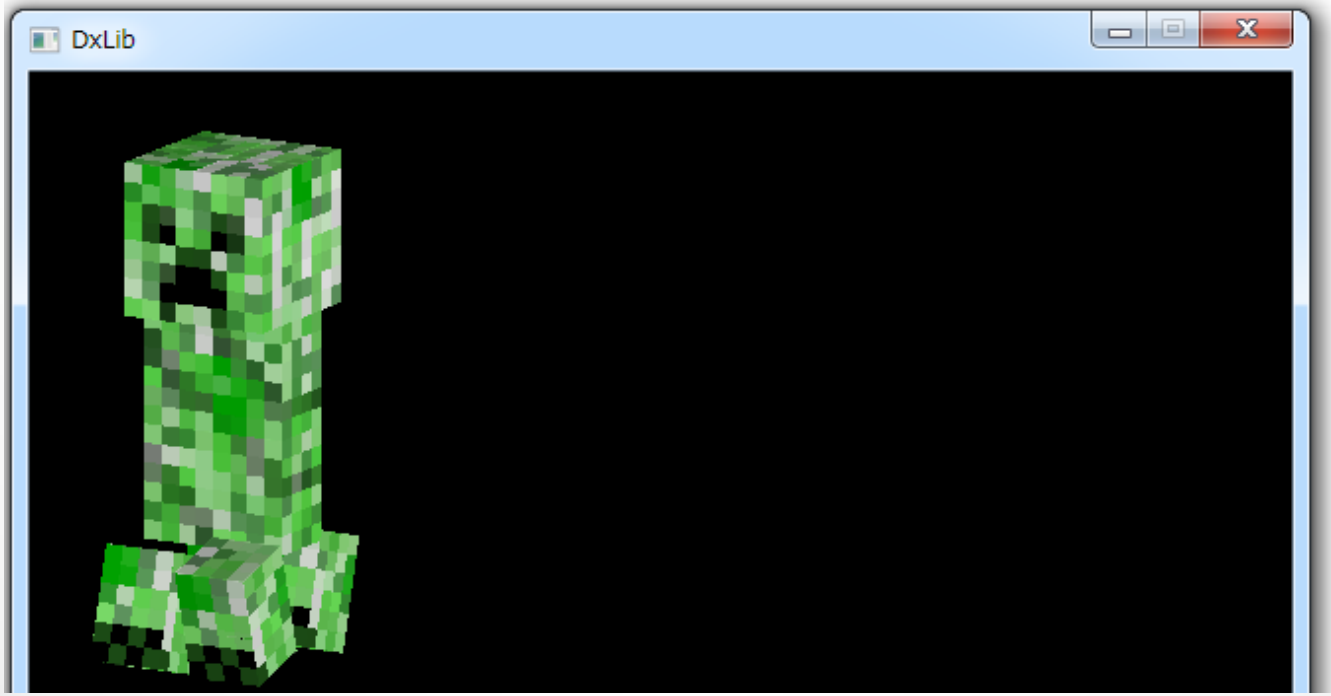
ソースコード

下準備

- ソースコードがあるフォルダに「Resource」という、画像や音楽を入れるためのフォルダを作る
- ネットなどから、pngの透過情報のある画像を拾ってくる
- picture.png と リネームし、Resourceフォルダに入れる

画像描画

```
1   #include "DxLib.h"
2
3   int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
4                       LPSTR lpCmdLine, int nCmdShow ){
5
6       ChangeWindowMode(TRUE);
7
8       DxLib_Init();
9
10      LoadGraphScreen( 30, 30, "Resource/picture.png", TRUE );
11
12      WaitKey();
13
14      DxLib_End();
15
16      return 0;
17  }
```



LoadGraphScreen関数

宣言	int LoadGraphScreen(int x, int y, char *GraphName, int TransFlag);
概略	画像ファイルを読み込み描画する
引数	x, y – 画像を描画する座標 GraphName – 描画する画像のファイルパス TransFlag – 透過色を入れるかどうか
戻り値	リファレンス参照
解説	char *GraphName は画像のファイルパスをstring型で指定する

- 直接画像を読み込んでいては効率が悪い
  (複数回描画すると、その都度ハードディスクから読み込むことになる)

- 一度、画像をメモリにロードして、ロードした画像を描画することで高速化が図れる

LoadGraph関数

宣言	int LoadGraph(char *FileName);
概略	画像ファイルをメモリに読み込む
引数	FileName – ロードする画像のファイルパス
戻り値	画像のデータハンドル
解説	char *FileName は画像のファイルパスをstring型で指定する

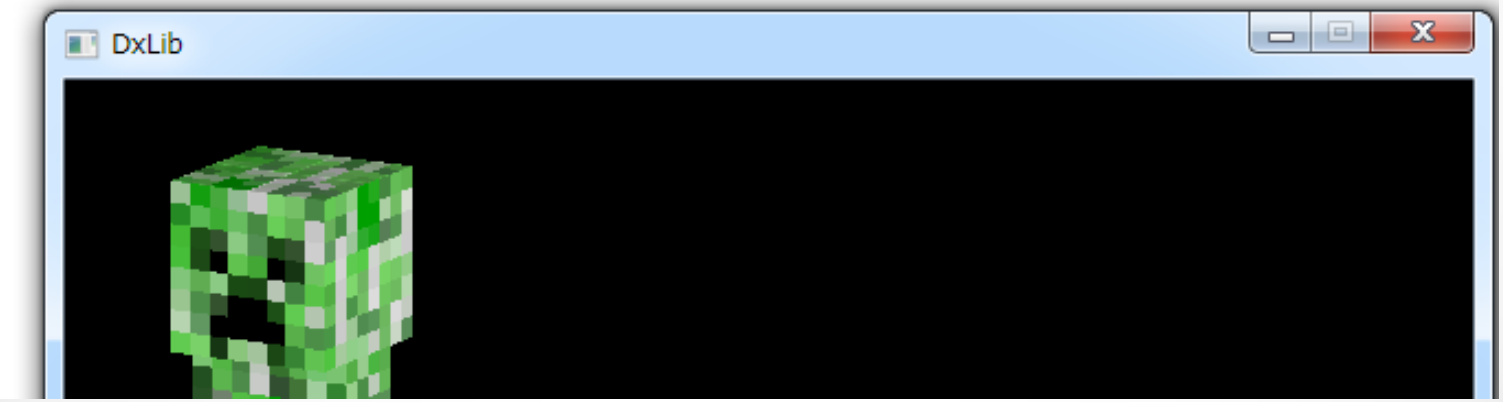
戻り値で画像のデータハンドルがint型で返ってくる。
描画するときに、このデータハンドルを描画関数に渡すと、
ロードした画像を描画してくれる

DrawGraph関数

宣言	int DrawGraph(int x, int y, int GrHandle, int TransFlag);
概略	画像ファイルを読み込み描画する
引数	x, y – 画像を描画する座標 GrHandle – 描画する画像のデータハンドル TransFlag – 透過色を入れるかどうか
戻り値	リファレンス参照
解説	LoadGraphで画像を読み込み、その戻り値を指定すると読み込んだ画像を描画する

画像描画

```
1   #include "DxLib.h"
2
3   int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
4                       LPSTR lpCmdLine, int nCmdShow ){
5
6       ChangeWindowMode(TRUE);
7
8       DxLib_Init();
9
10      int handle = LoadGraph("Resource/picture.png");
11
12      DrawGraph(30, 30, handle, TRUE);
13
14      WaitKey();
15
16      DxLib_End();
17
18      return 0;
19  }
```



画像の代表的な拡張子

- jpg(jpeg)
  - **非可逆圧縮**(元に戻らない)の画像フォーマット
  - ゲームの背景などに使用

- png
  - **可逆圧縮**(劣化のない)のビットマップ画像フォーマット
  - **透過情報**を扱うことができる
  - ゲームのキャラクターなどに使用

- gif
  - ~~岐阜県民専用画像フォーマット~~

まとめ

- LoadGraphScreenは直接画像を表示する
- LoadGraphで画像をメモリにロードし、DrawGraphでLoadGraphの戻り値の値を指定して描画することで高速に描画することができる
- 描画したい画像をソースコードのあるディレクトリに置くことを忘れない
- 画像ファイルには種類がある(png, jpgなど)

キー入力

- WaitKey()で何か押されたら終了 ではなく、
  ESCキーが押されたら終了にしてみる

ソースコード

```
1   #include "DxLib.h"
2
3   int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
4                       LPSTR lpCmdLine, int nCmdShow ){
5       ChangeWindowMode(TRUE);
6       DxLib_Init();
7       int handle = LoadGraph("Resource/picture.png");
8       DrawGraph(30, 30, handle, TRUE);
9
10      while(true) {
11          if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;
12      }
13
14      DxLib_End();
15
16      return 0;
17  }
```

CheckHitKey関数

宣言	int CheckHitKey(int KeyCode)
概略	特定のキーが入力されているか調べる
引数	KeyCode – 入力状態を取得するキーコード
戻り値	0 : 押されていない / 1 : 押されている
解説	引数には、DxLibで定義されている識別子を使用する 例えば、Aが押されているときに処理をしたい場合、 if(CheckHitKey(KEY_INPUT_A) == 1) 処理(); と、する・

- 無限ループにして、キーの入力をチェックして押されたらbreakするようになった
- 無限ループにする→まずい
- プログラムとWindowsとで適度なタイミングで応答を取り合うようにしなければならない

ProcessMessage関数

宣言	int ProcessMessage()
概略	ウィンドウのメッセージを処理する
引数	なし
戻り値	0 : 成功 / -1 : エラー発生
解説	定期的に呼び出す必要がある エラーが起きたらプログラムを終了させる必要がある

☑ProcessMessage関数を使わずに実行し、
ウィンドウの×ボタンを押して見る
(終了するときはVisualStudioの■ボタンを押す)

ソースコード

```
1   #include "DxLib.h"
2
3   int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
4                       LPSTR lpCmdLine, int nCmdShow ){
5       ChangeWindowMode(TRUE);
6       DxLib_Init();
7       int handle = LoadGraph("Resource/picture.png");
8       DrawGraph(30, 30, handle, TRUE);
9
10      while(true) {
11          if(ProcessMessage() == -1) break;
12          if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;
13          /*
14              メインループ
15          */
16      }
17      return 0;
18  }
```

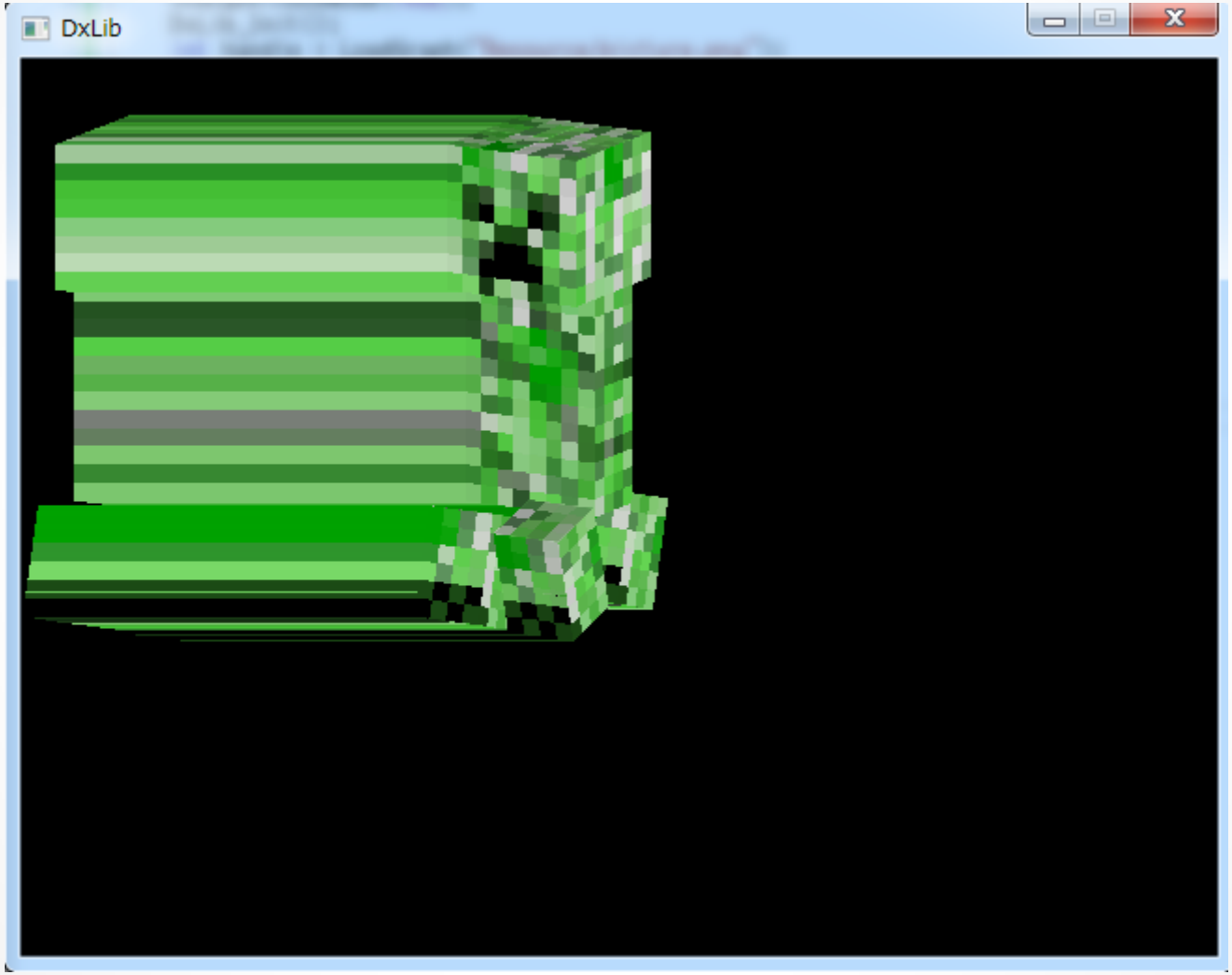
アニメーションさせてみる

- 変数 int x を用意する
- while文の中で、x の値を増やしていく
- DrawGraphの第一引数に x を指定する
  例) DrawGraph(x, 30, handle, TRUE);
- while文の終わりにSleep(10); を入力しておく
  (Sleep関数は指定された秒数待機する
  この場合だと、ループの度に10msec待機)

ソースコード

```
1   #include "DxLib.h"
2
3   int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
4                       LPSTR lpCmdLine, int nCmdShow ){
5       ChangeWindowMode(TRUE);
6       DxLib_Init();
7       int handle = LoadGraph("Resource/picture.png");
8
9       int x = 0;
10
11      while(true) {
12          if(ProcessMessage() == -1) break;
13          if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;
14
15          DrawGraph(x, 30, handle, TRUE);
16          x += 1;
17
18          Sleep(10);
19      }
20      return 0;
21  }
```

実行結果



実行結果

！　？

- 描画したら消さなければならない
- 移動 → 描画 → 消す →移動 → 描画 → 消す → ・・・・でアニメーションを作っていく
- 動画サイトでよく見る黒板を使ったアニメーションがいい例

ClearDrawScreen関数

宣言	int ClearDrawScreen()
概略	画面に描かれたものを消去する
引数	なし
戻り値	省略
解説	描画関数で描画されたグラフィクスを消して、画面を初期化する

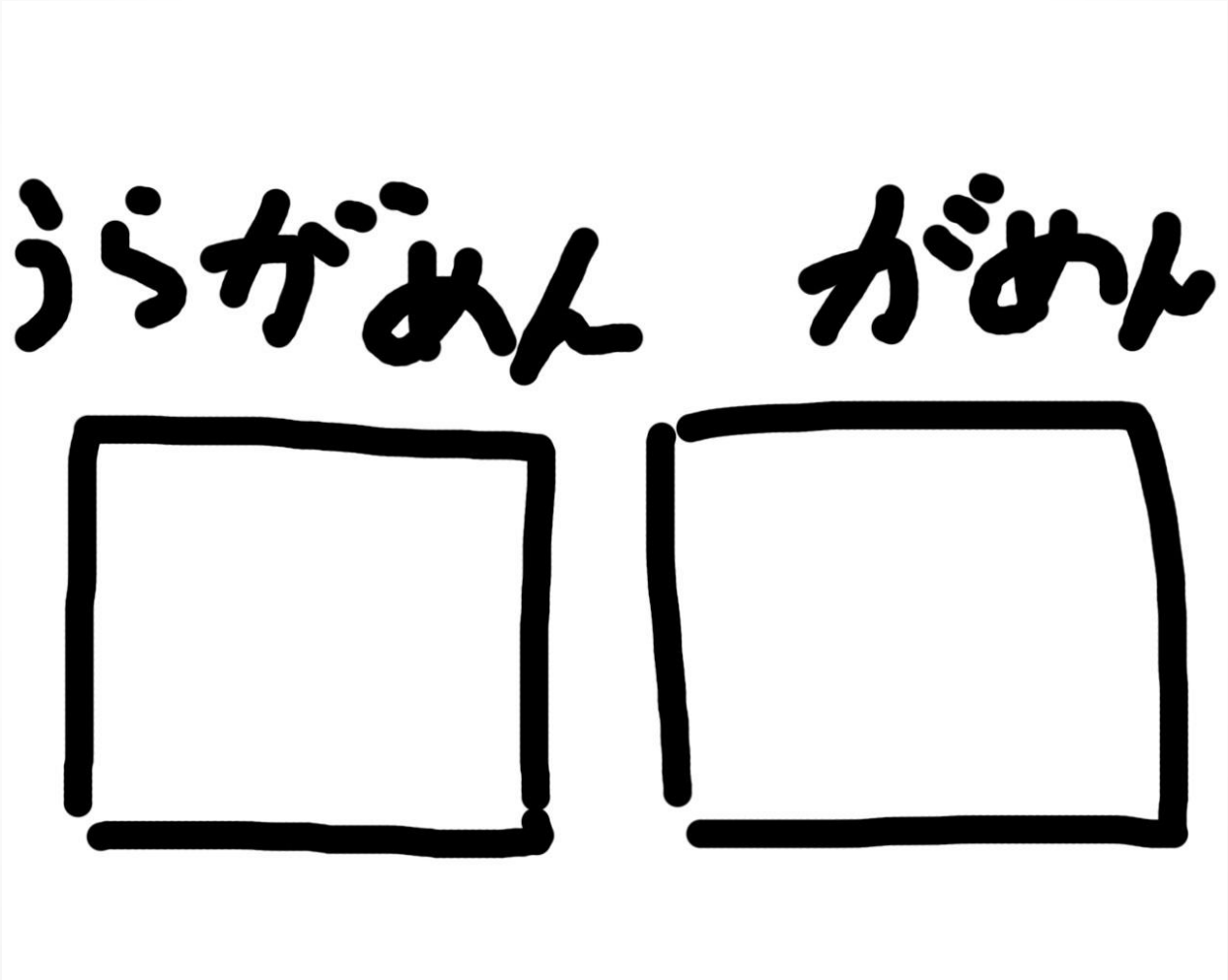
☑while文のProcessMessage関数、CheckHitKey関数の次に呼び出すようにし、実行する

実行結果

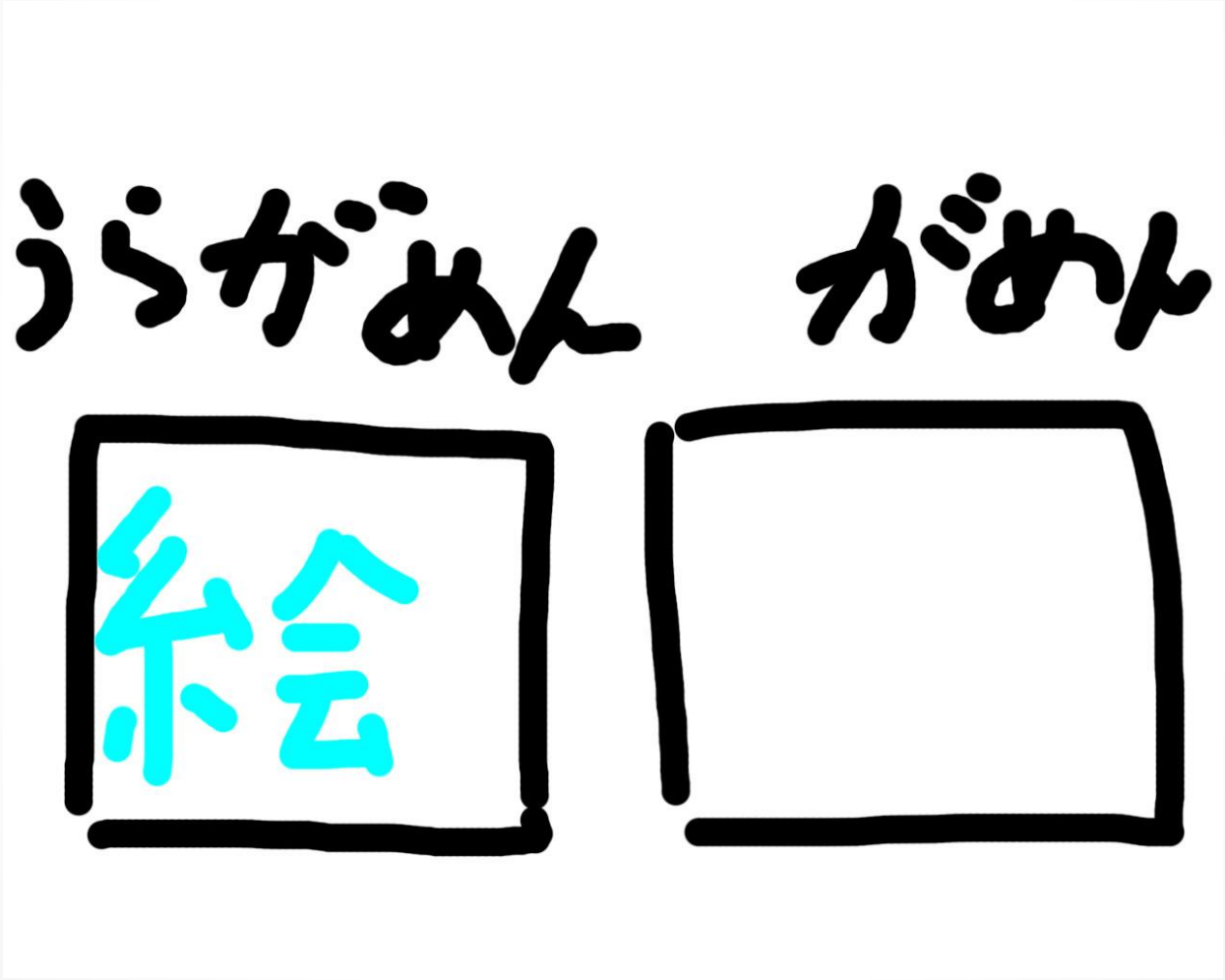
ぎこちない

ダブルバッファリング

ダブルバッファリング



ダブルバッファリング



ダブルバッファリング

ダブルバッファリング

ダブルバッファリング

描画しながら、次のフレームの準備ができる
なめらかに！

SetDrawScreen関数

宣言	int SetDrawScreen(int DrawScreen)
概略	描画先グラフィクス領域の指定
引数	DrawScreen : 描画対象のグラフィクス領域
戻り値	省略
解説	描画を裏画像に SetDrawScreen(DX_SCREEN_BACK);

裏画面の作成

ScreenFlip関数

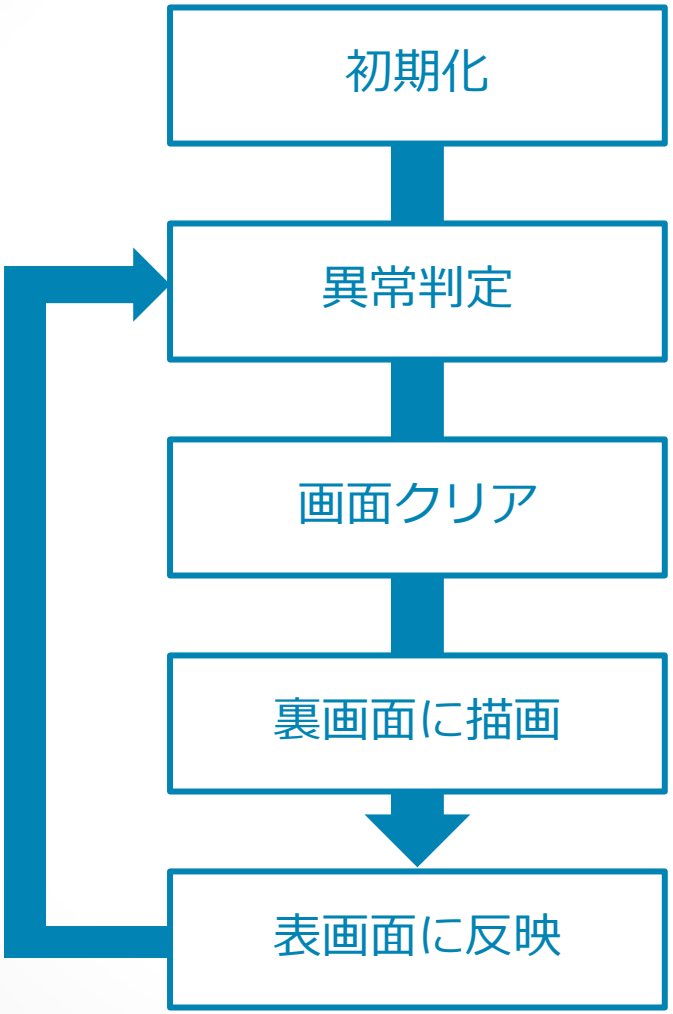
宣言	int ScreenFlip()
概略	裏画面に描画したものを表画面に反映する
引数	DrawScreen : 描画対象のグラフィクス領域
戻り値	省略
解説	

裏画面に描画したものを表画面に反映

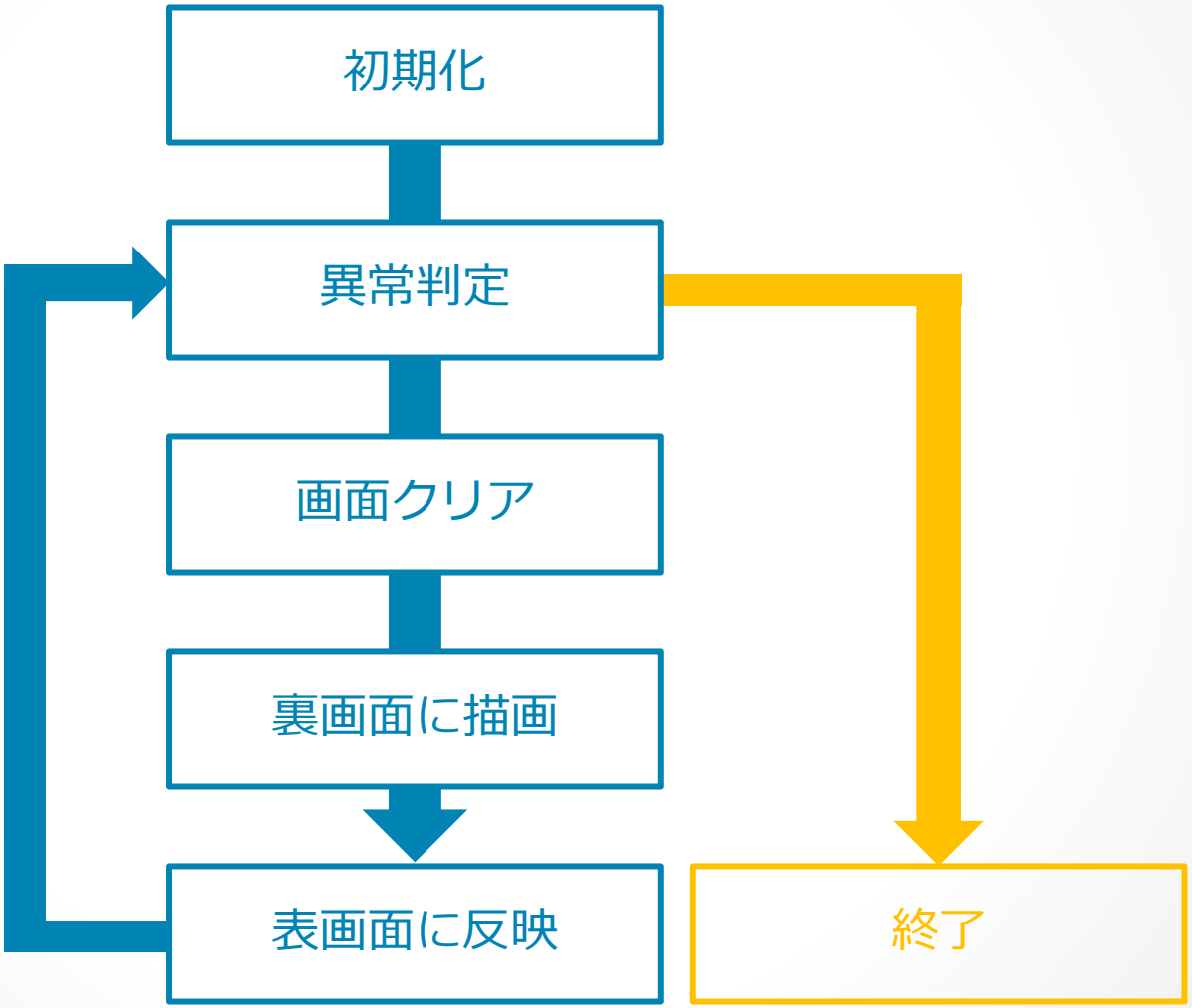
ソースコード

```
1   #include "DxLib.h"
2
3   int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
4                       LPSTR lpCmdLine, int nCmdShow ){
5       ChangeWindowMode(TRUE);
6       SetDrawScreen(DX_SCREEN_BACK);
7       DxLib_Init();
8       int handle = LoadGraph("Resource/picture.png");
9
10      int x = 0;
11
12      while(true) {
13          if(ProcessMessage() == -1) break;
14          if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;
15
16          ClearDrawScreen();
17
18          DrawGraph(x, 30, handle, TRUE);
19          x += 1;
20
21          ScreenFlip();
22      }
23      return 0;
24  }
```

まとめ



まとめ



課題

- 画像を3つ描画する
- それぞれ、描画する x 座標か y 座標を変数にしておき、while 文の中で値を変化させることでアニメーションさせてみる。
- 1つ目の画像　→　横移動(右から左へ、画面外に出そうになったら、左から右へ移動)
- 2つめの画像　→　縦移動(上から下へ、画面外に出そうになったら、下から上へ移動)
- 3つ目の画像　→　DxLibで乱数を取得し(リファレンスから調べること)　x , y に乱数の値を指定する。

課題

- 3番補足

- 高速移動することになるので、while文の度にランダムの値を取るのでなく、30回繰り返すごとに取得する

```
count += 1;
if(count % 30 == 0) {
    x = GetRand(600);
    y = GetRand(400);
}

DrawGraph(x, y, handle, TRUE);
```