DXライブラリを用いた
ブロック崩し作成 part.1

DrawCircle関数
GetColor関数
ClearDrawScreen関数
SetDrawScreen関数
ScreenFlip関数
を用いてボールを描画してみる。

DXライブラリとは？

- C++上で動く関数群

- とにかく画像の描画が簡単！

  → 初心者にも簡単にゲームを作れるオススメのライブラリ

- 処理速度が早い！

  → シューティングなどの計算が膨大なゲームでも
  中々の速度で動いてくれる。

ソースの最初の構成

main.cpp*

(グローバル スコープ)

```
#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 )
{
    ChangeWindowMode(TRUE);//ウィンドウモードで起動

    DxLib_Init();   // ＤＸライブラリ初期化処理

    //初期化処理はここに書く！！


    while(ProcessMessage()==0){

        //メインループの動作はここに書く！！

    }

    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
}
```

- **初期化処理とは？**

  ブロック崩しの初期化で言えば、
  ボールとブロックを作ってやる部分がここになる。
- **Whileループ内で行う処理とは？**

  主にボールがどのように動くか等、
  ゲームのメイン部分の命令をここに書くことになる。
- **whileの条件式のProcessMessage()って？**

  簡単に言うと正常に動いている時のみ0
  異常な動作やウィンドウ左上の×を押すと-1が返ってくる関数
  つまり、異常な動作やウィンドウを閉じる動作をしたとき、
  ループを抜け出すようにwhileの条件式に書いている。

色を取得するGetColor関数

- GetColor(int Red,int Green,int Blue)

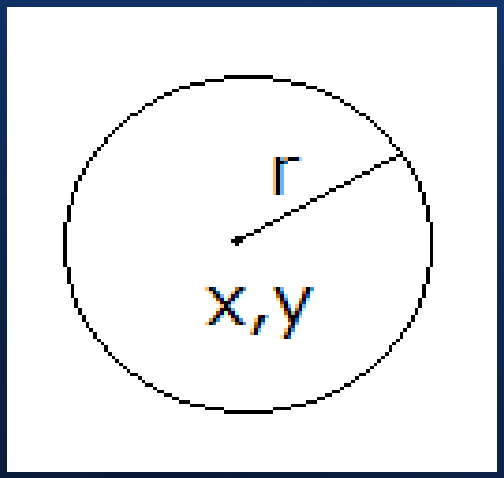
  いわゆる光の三原色である。
  使い方としてはRed,Green,Blueを
  指定してやればその色となる。

  ちなみにRed,Green,Blueが受け取れる値は0～255まで。

  例えば、GetColor(255,255,255)とすれば白になるし、
  GetColor(0,0,255)とすれば青になる。

丸を描画するDrawCircle関数

- DrawCircle(int x,int y,int r,int color,int FillFlag);



引数のx,yは円の中心、rは円の半径となる。

colorに入る部分により、円の色決定。
GetColor(int R,int G,int B)関数を用いるとよい。

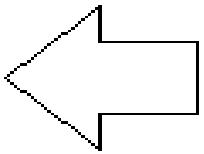
FillFlagは
TRUEの時　円の内部も塗りつぶし
FALSEの時　塗りつぶしはなしとなる。
左図はFALSEの時ということになる

DrawCircle,GetColorの例

main.cpp*

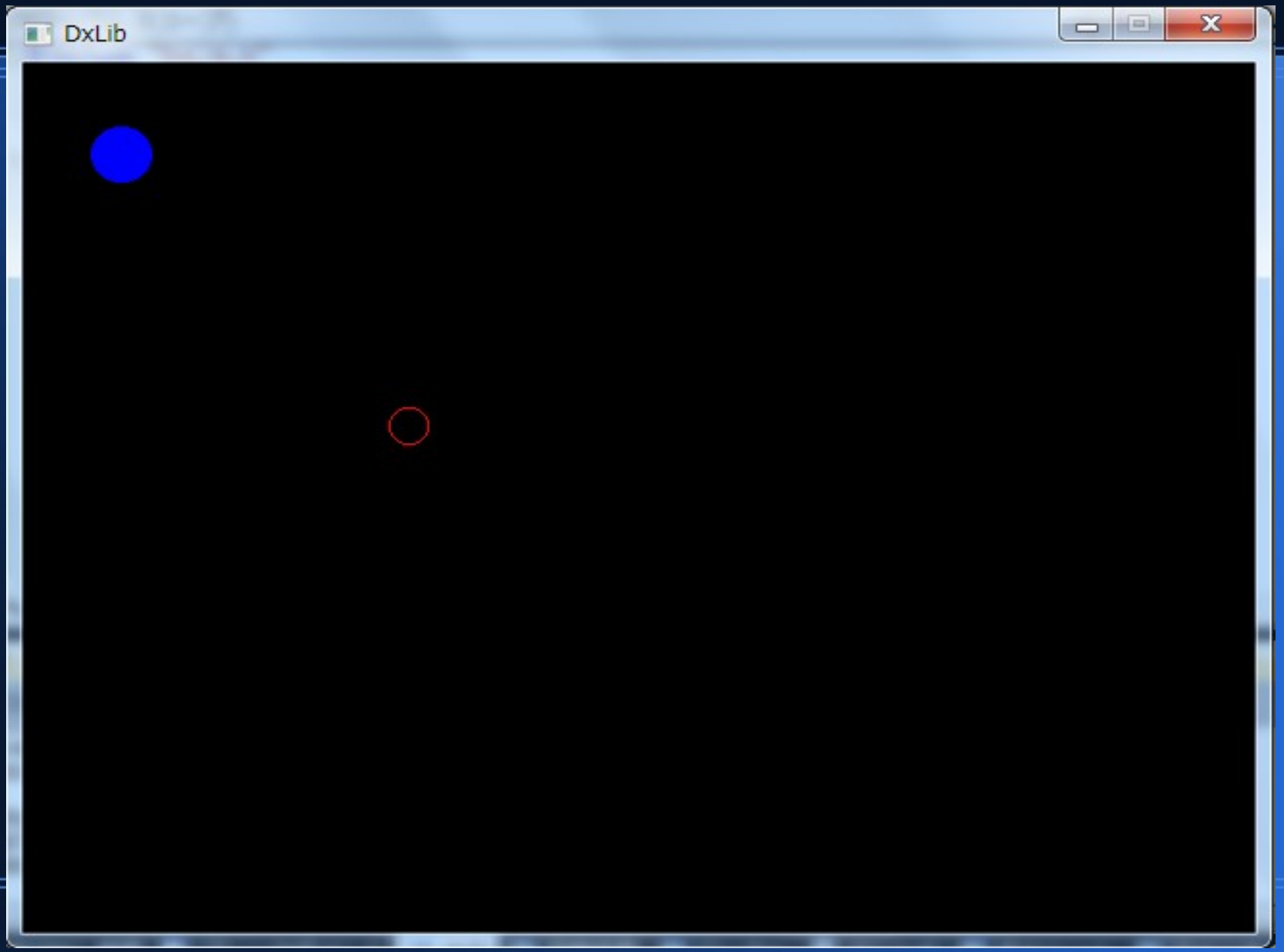
(グローバル スコープ)

```
#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 )
{
    ChangeWindowMode(TRUE);//ウィンドウモードで起動

    DxLib_Init();   // ＤＸライブラリ初期化処理

    //初期化処理はここに書く！！


    while(ProcessMessage()==0){

        //メインループの動作はここに書く！！
        DrawCircle(50,50,15,GetColor(0,0,255),TRUE);
        DrawCircle(200,200,10,GetColor(255,0,0),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
}
```



追加部分！

実行結果



追加部分の説明

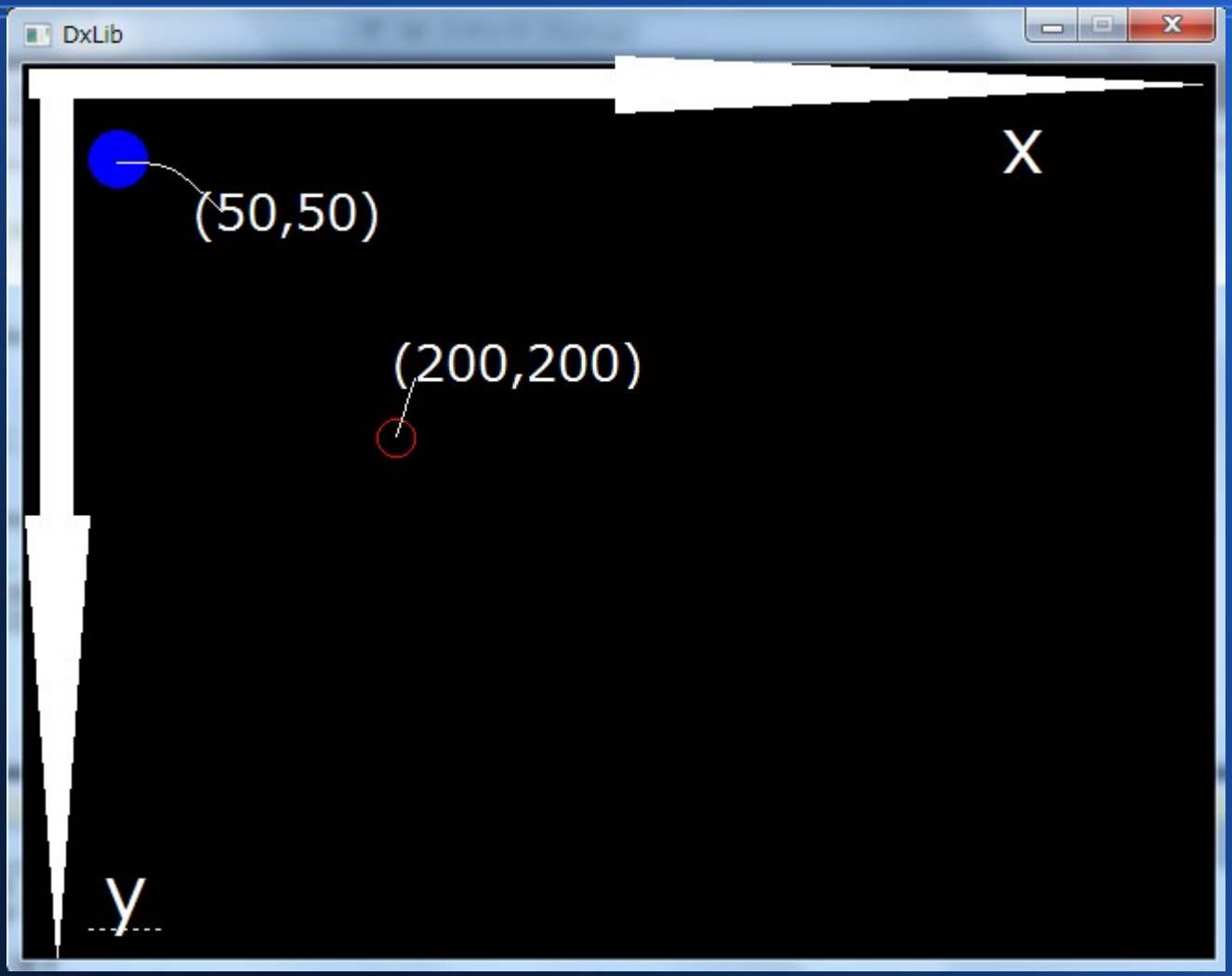
- DrawCircle(50,50,15,GetColor(0,0,255),TRUE);

  最初の50,50にてx,y座標が決定。15にて半径のrが決定。
  GetColor(0,0,255)にて青が選択される。
  FillFlagがTRUEなので塗りつぶし。

- DrawCircle(200,200,10,
  GetColor(255,0,0),FALSE);

  最初の200,200でx,y座標決定。10が円の半径。
  GetColor(255,0,0)にて赤が選択される。
  FALSEにて塗りつぶされない。

ウィンドウのx,y座標について



このように
右にいくにつれて
X座標が増えて

下に行くにつれて
Y座標が増える
ことになる。

ボールを動かしてみる！

- ループ外でint x = 50を設定。

- ループ内でx++;として
- DrawCircle(x,50,15,GetColor(0,0,255),TRUE);
- とすればどんどんxが増えてくれるので
- ボールが動いてるように見えるはず！

ソースコード例

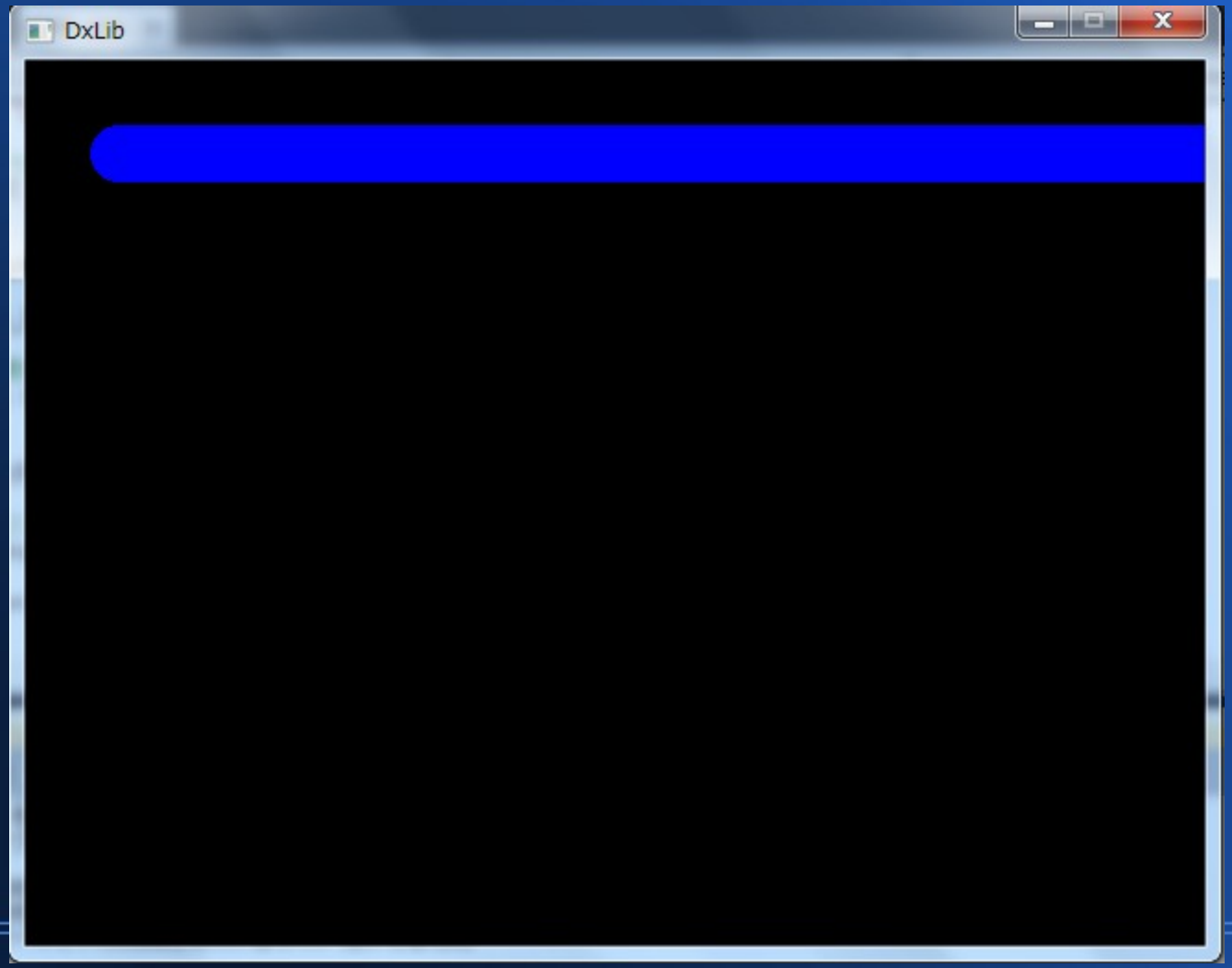
main.cpp*

(グローバル スコープ)

```
#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 )
{
    ChangeWindowMode(TRUE);//ウィンドウモードで起動

    DxLib_Init();   // ＤＸライブラリ初期化処理

    //初期化処理はここに書く！！

    int x = 50;//ボールのx座標を50に設定。


    while(ProcessMessage()==0){

        //メインループの動作はここに書く！！

        //ボールの描画
        DrawCircle(x,50,15,GetColor(0,0,255),TRUE);

        x++;//ボールのx座標増やす。

    }

    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
}
```

追加部分

実行結果



何故このようになるか
というと、
最初のループで
50,50に描画
その次に51,50に描画、
その次に52,50と描画
しているのだが
前に描いたものを
消す命令をして
いないからである。

つまり消す処理を
いれてやれば動くはず！

ClearDrawScreen関数

- ClearDrawScreen()関数

- 前に描いたものを消す関数
  これを用いることにより、
  ずっと描き続けられるのを防ぐ。

ソース例

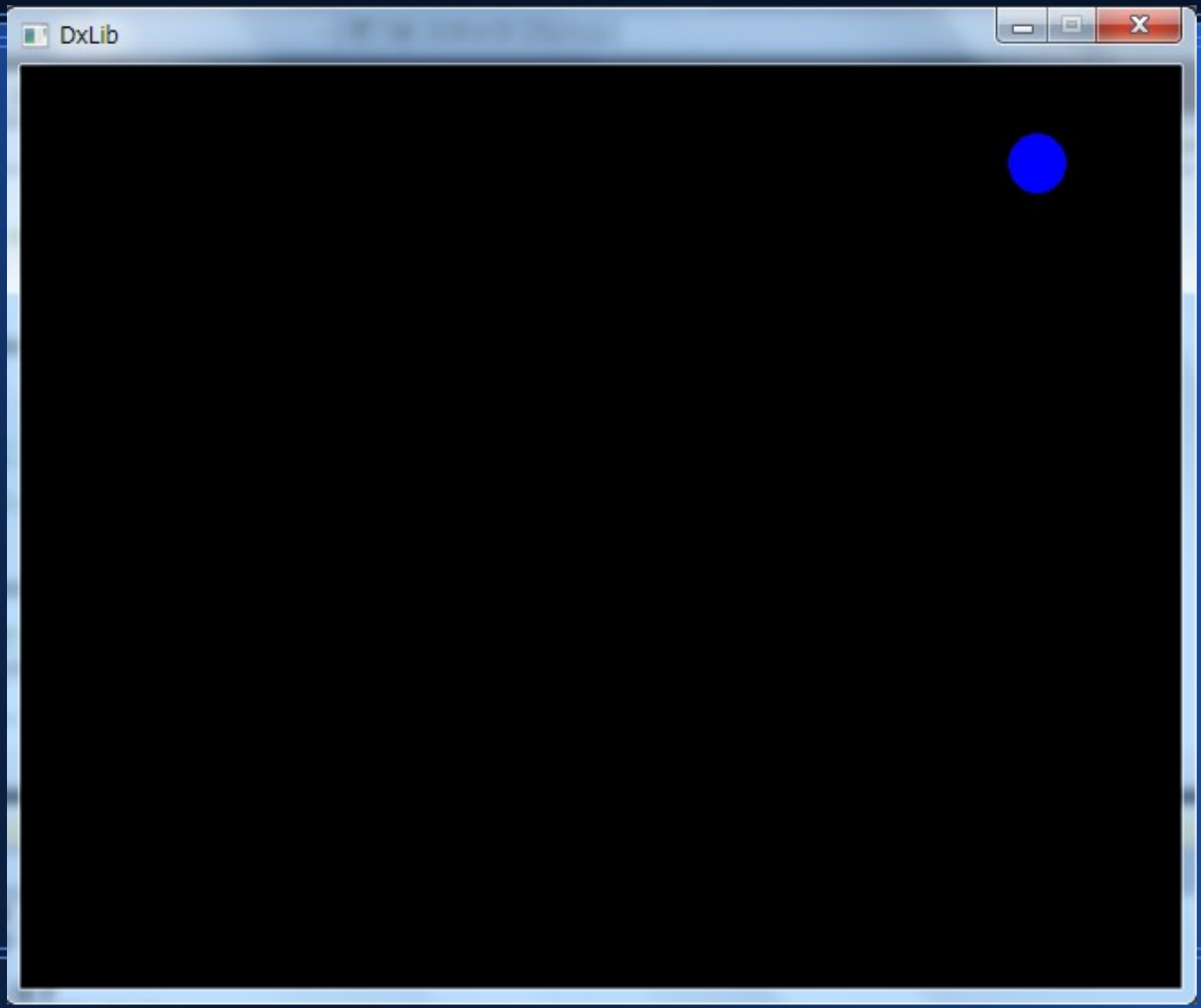
main.cpp*

(グローバル スコープ)

```
#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 )
{
    ChangeWindowMode(TRUE);//ウィンドウモードで起動

    DxLib_Init();   // ＤＸライブラリ初期化処理

    //初期化処理はここに書く！！

    int x = 50;//ボールのx座標を50に設定。


    while(ProcessMessage()==0){

        //描画されているものを消す。
        ClearDrawScreen();

        //メインループの動作はここに書く！！

        //ボールの描画
        DrawCircle(x,50,15,GetColor(0,0,255),TRUE);

        x++;//ボールのx座標増やす。

    }

    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
}
```

追加部分

実行結果



確かに移動したが、
とびとびで移動してるように
見えただろう。

このように見える理由は
今回の処理では
描画→消去→描画→消去
とやっていたのがわかる。

つまり、人間の目に消去の
瞬間が見えてしまい、
とびとびに移動してるように
見えたのだ。

SetDrawScreen関数

- 実はDXライブラリには描画するボードみたいなのが2つ（表と裏）準備されている。
- 普段、画面として出力されているのは表。
- SetDrawScreen(int DrawScreen);
- DrawScreenにて表に描画するか、裏に描画するかを決定する。
- DrawScreenがDX_SCREEN_FRONTを受け取った時は表
- DX_SCREEN_BACKを受け取った時は裏

ScreenFlip関数

- ScreenFlip()関数は裏に描画されたものを表に反映するものである。
- つまり画像の描画などを全て裏で行い、描画が完了したら表に反映する。
- このようにすることで、消去の瞬間を人間の目に見せない手段をとる。
- よって追加事項は初期化の部分に
- SetDrawScreen(DX_SCREEN_BACK)と
- ループ内にScreenFlip()関数。

ソース例

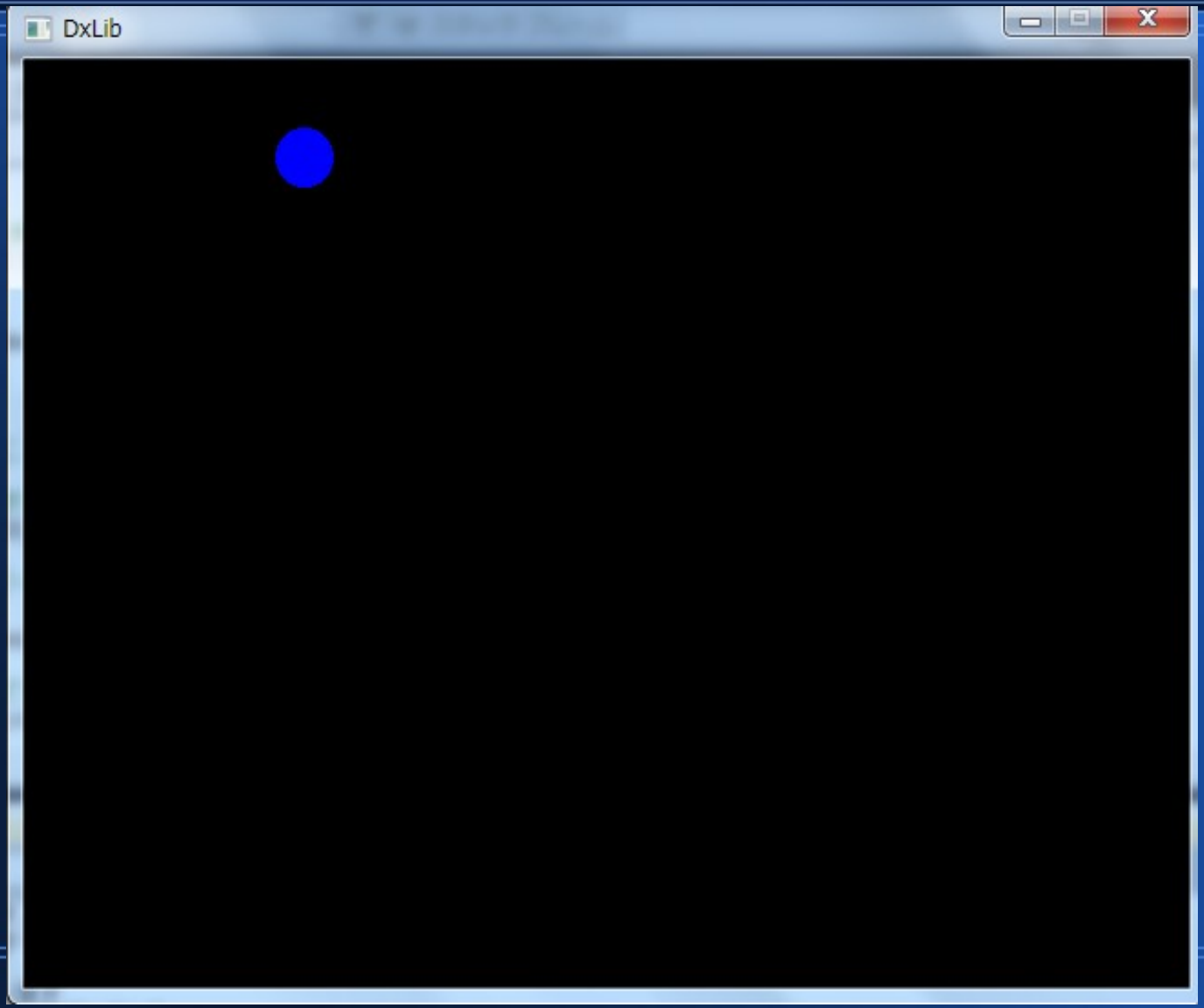
main.cpp*

(グローバル スコープ)

```
#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 )
{
    ChangeWindowMode(TRUE);//ウィンドウモードで起動

    DxLib_Init();   // ＤＸライブラリ初期化処理

    //初期化処理はここに書く！！

    //裏画面に描画することを決定。
    SetDrawScreen(DX_SCREEN_BACK);

    int x = 50;//ボールのx座標を50に設定。

    while(ProcessMessage()==0){

        //描画されているものを消す。
        ClearDrawScreen();

        //メインループの動作はここに書く！！

        //ボールの描画
        DrawCircle(x,50,15,GetColor(0,0,255),TRUE);

        x++;//ボールのx座標増やす。

        //裏画面の描画状態を表に反映
        ScreenFlip();
    }

    WaitKey() ;             // キー入力待ち
```

追加事項

実行結果



今回はぬるぬる
動いているのが
わかるだろう。

ちなみに
SetDrawScreenにて
裏画面を選んだ場合
ClearDrawScreen
も裏にしか
影響しないので
人間の目には消去の瞬間
が見えていないのである。

何かを描画したい時すること。

- 何かを描画する時は
- ループ外に
  SetDrawScreen(DX_SCREEN_BACK)
- ループ内に
- ClearDrawScreen();とScreenFlip();
  を忘れない！！

ソース例

main.cpp*

(グローバル スコープ)

```
#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 )
{
    ChangeWindowMode(TRUE);//ウィンドウモードで起動

    DxLib_Init();   // ＤＸライブラリ初期化処理

    //初期化処理はここに書く！！

    //裏画面に描画することを決定。
    SetDrawScreen(DX_SCREEN_BACK);

    while(ProcessMessage()==0){

        //描画されているものを消す。
        ClearDrawScreen();

        //メインループの動作はここに書く！！

        //裏画面の描画状態を表に反映
        ScreenFlip();
    }

    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
}
```

DXライブラリで
ゲームを作るときは
大抵ここからスタートする。

なのでこれは覚える必要なく
コピペですましてしまっても
構わないだろう。

まとめ

- DrawCircle(int x,int y,int r,int color,int FillFlag)
  により、円の描画。
- GetColor(int Red,int Blue,int Green)
  により、色の取得。
- **描画する時は**
  SetDrawScreen**と**ClearDrawScreen,ScreenFlip
  を用いる。