DXライブラリを用いて
ブロック崩しを作るpart.2

DrawBoxを用いたブロックの作成

DrawBox関数

- DrawBox(int x1,int y1,int x2,int y2,
  int color,int FillFlag);
- **四角形を描画する関数**

x1,y1にて四角形の左上の座標を決定
x2,y2にて四角形の右下の座標を決定
Colorの部分はDrawCircleと同じ
GetColorを用いるとよい
FillFlagも同様、TRUEなら塗りつぶし、
FALSEなら外側のみとなる。

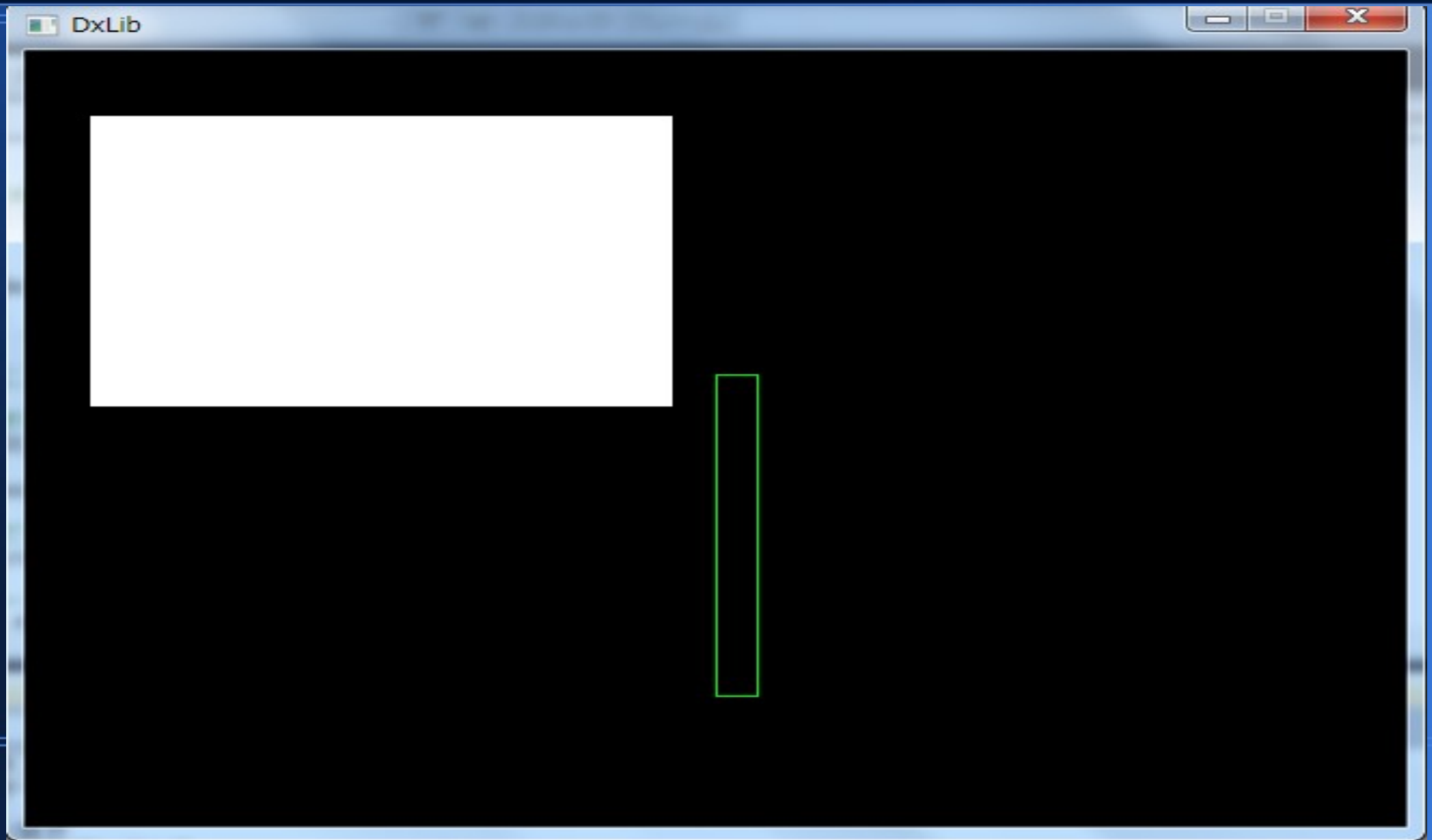
ソース例

main.cpp

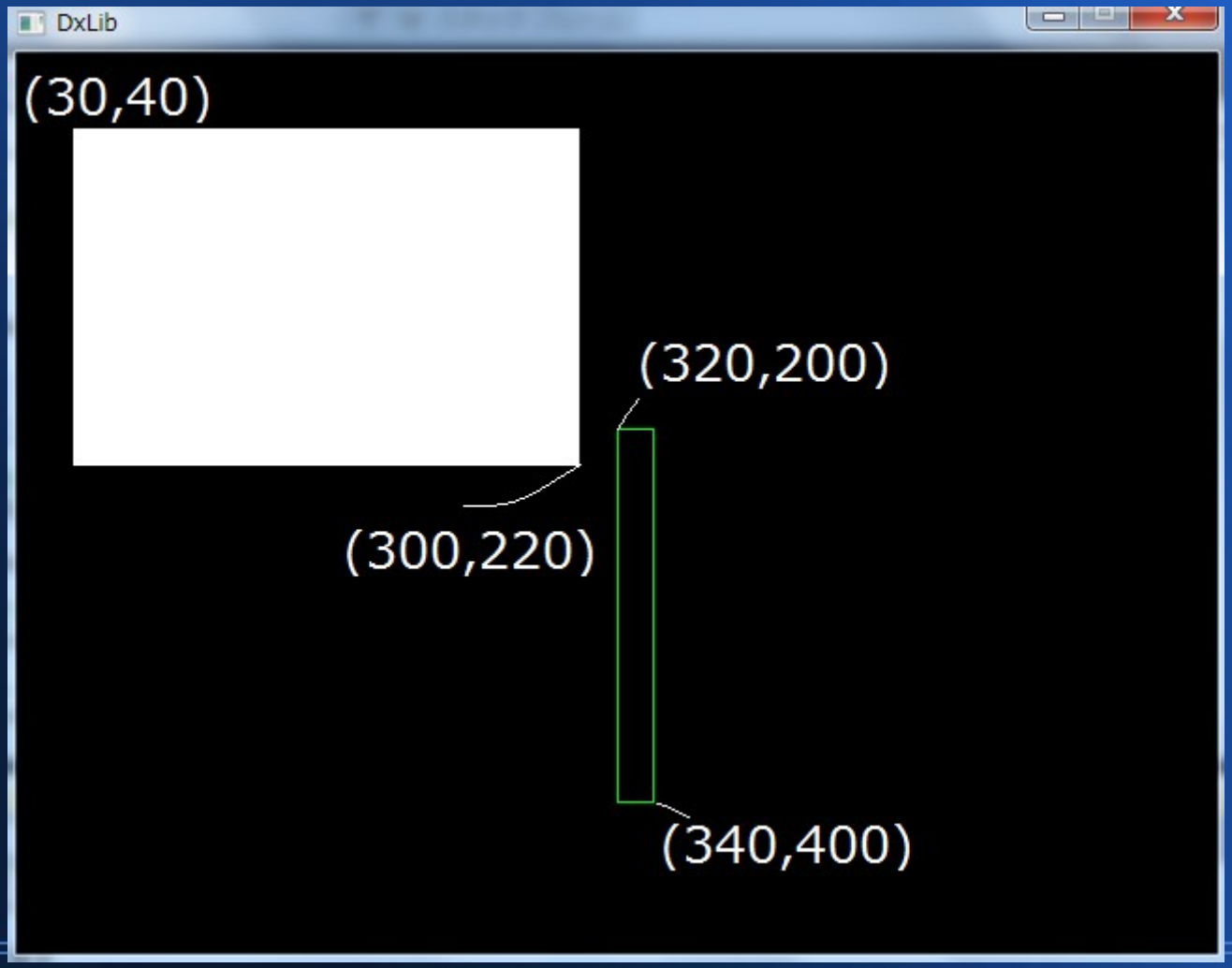
(グローバル スコープ)

```
#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！！
        DrawBox(30,40,300,220,GetColor(255,255,255),TRUE);
        DrawBox(320,200,340,400,GetColor(0,255,0),FALSE);

        //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
}
```

実行結果



実行結果



使い方はほぼ
DrawCircleと
同じである。

FillFlagがTRUEなら
塗りつぶし（白の箱）

FillFlagがFALSEなら
外側のみとなる（緑）

箱も移動させてみる。

- もちろんpart1でやった球の移動のように
  箱を移動させることも可能である。
- 注意事項としてはx方向に移動させる時
  DrawBox(x,50,200,120,・・・）のように
  左上の点のみ移動させないことである。
- これをやってしまうと右下の点は動かないので
  正しく箱は動いてくれない。

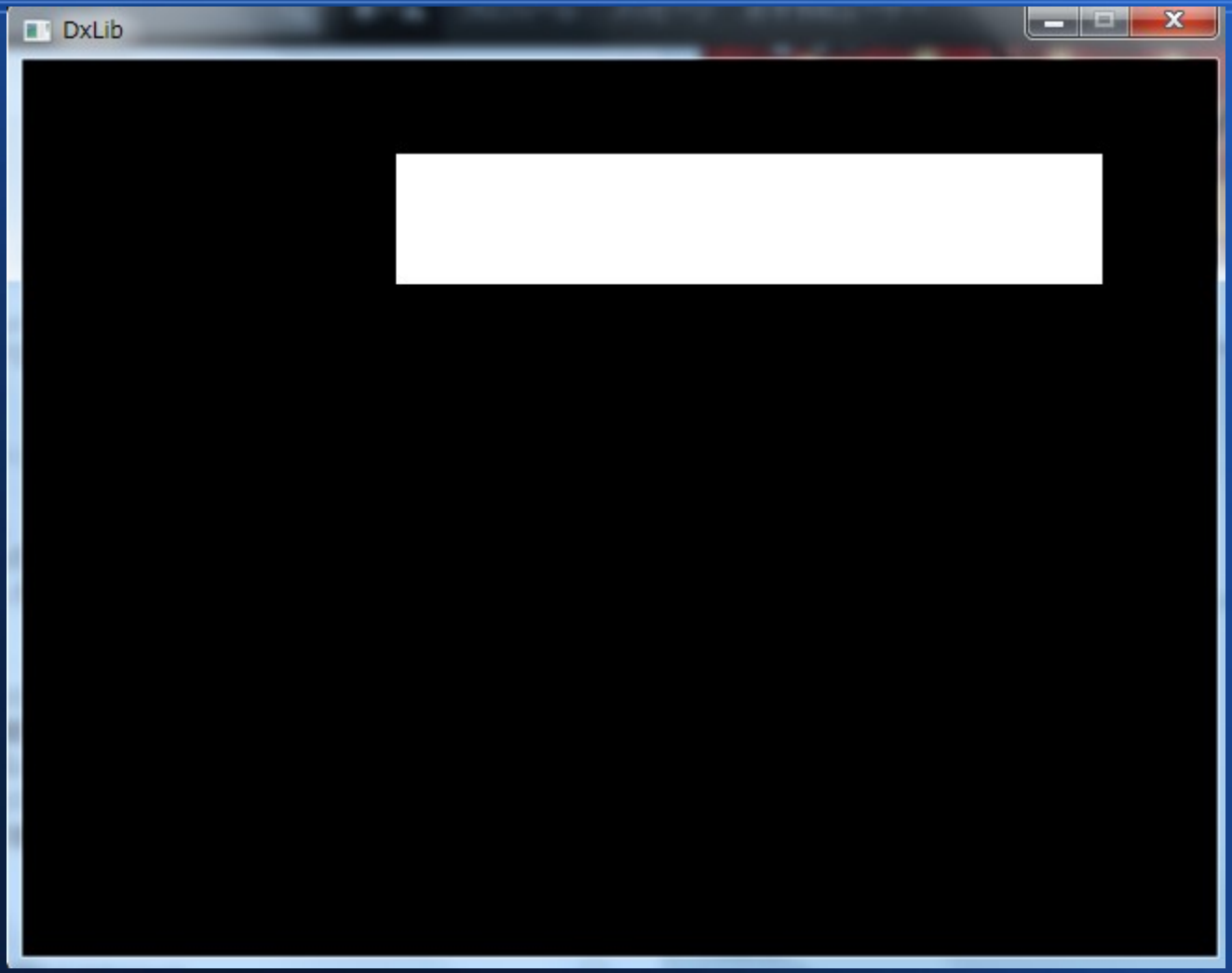
ソース例

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);

    //箱の左上の初期座標
    int x = 50;

    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！！
        //箱の描画
        DrawBox(x,50,200,120,GetColor(255,255,255),TRUE);

        //箱のx座標を動かす
        x++;

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

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

実行結果



このように、
左上の座標が動き
右下の座標が
固定されるので
四角形の形が
変わってしまうことになる。

改善策を考えて
実際に実行してみてから
次のページを見るのを
オススメする。

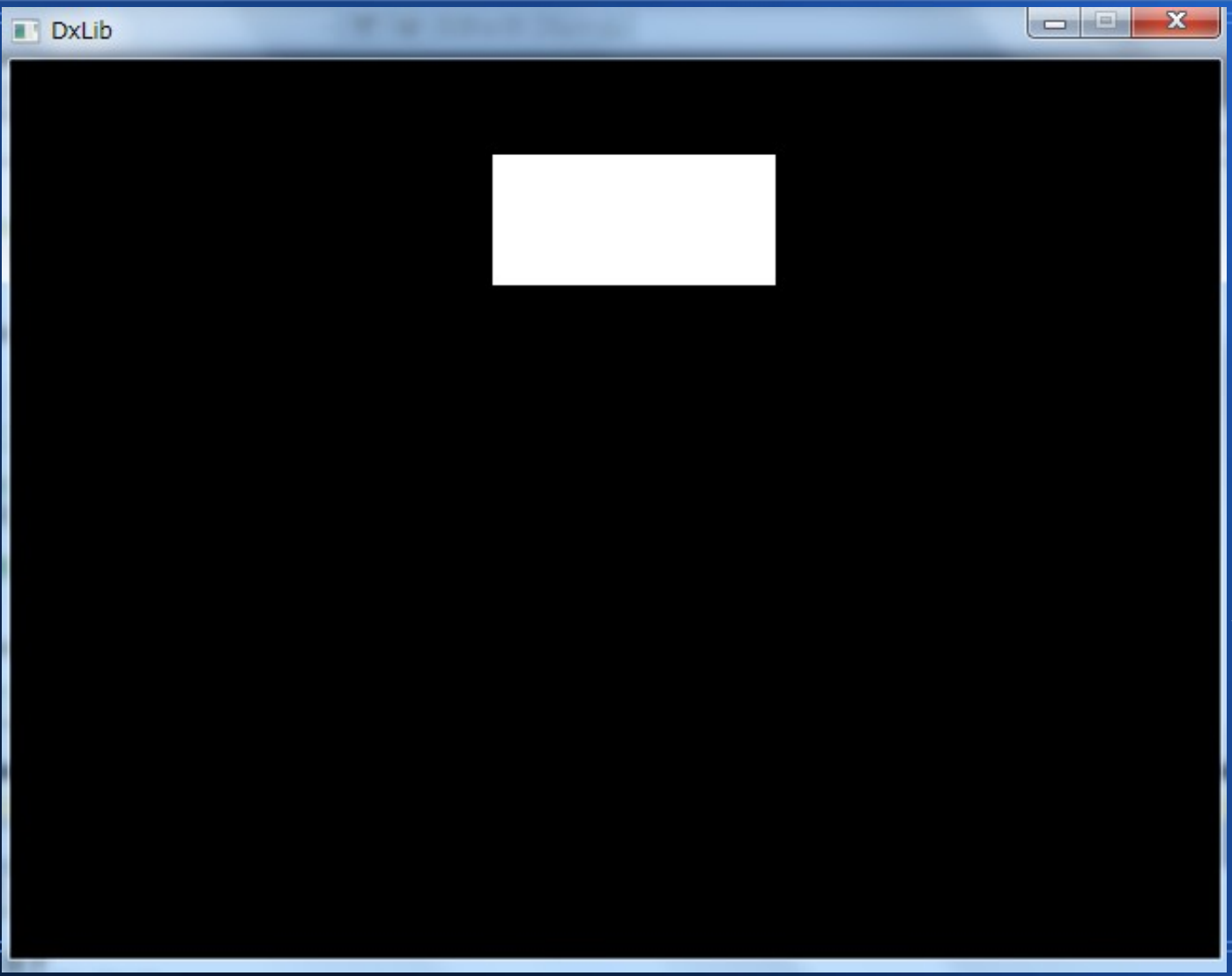
改善したソース例

```
#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);

    //x1は箱の左上、x2は箱の右下、vxはx方向の速度
    int x1=50,x2=200,vx=1;

    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！！
        //箱の描画
        DrawBox(x1,50,x2,120,GetColor(255,255,255),TRUE);

        //箱のx座標を動かす
        x1+=vx;
        x2+=vx;

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

左上、右下のx座標を
それぞれ指定し
速度だけ両方共動かす、
という手法をとった。

Y方向の速度もつくり
上下に移動させることも
可能なので
ぜひやってみるとよい。

実行結果



このように、
左上と右下の座標
両方動かすことにより
四角形を動かすことに
成功した。

少し改良して
Y方向にも動く四角形を
作って見るとよい。

forループを用いて
大量にブロックを作る

- ブロック崩しのブロックはやはり大量にある。
- これを用いてまずは横に
  ブロックを作って見ることとする。
- ブロックの横の数を簡単に減らすために
  #define BLOCK_X_NUM
  というマクロを設定しておく。

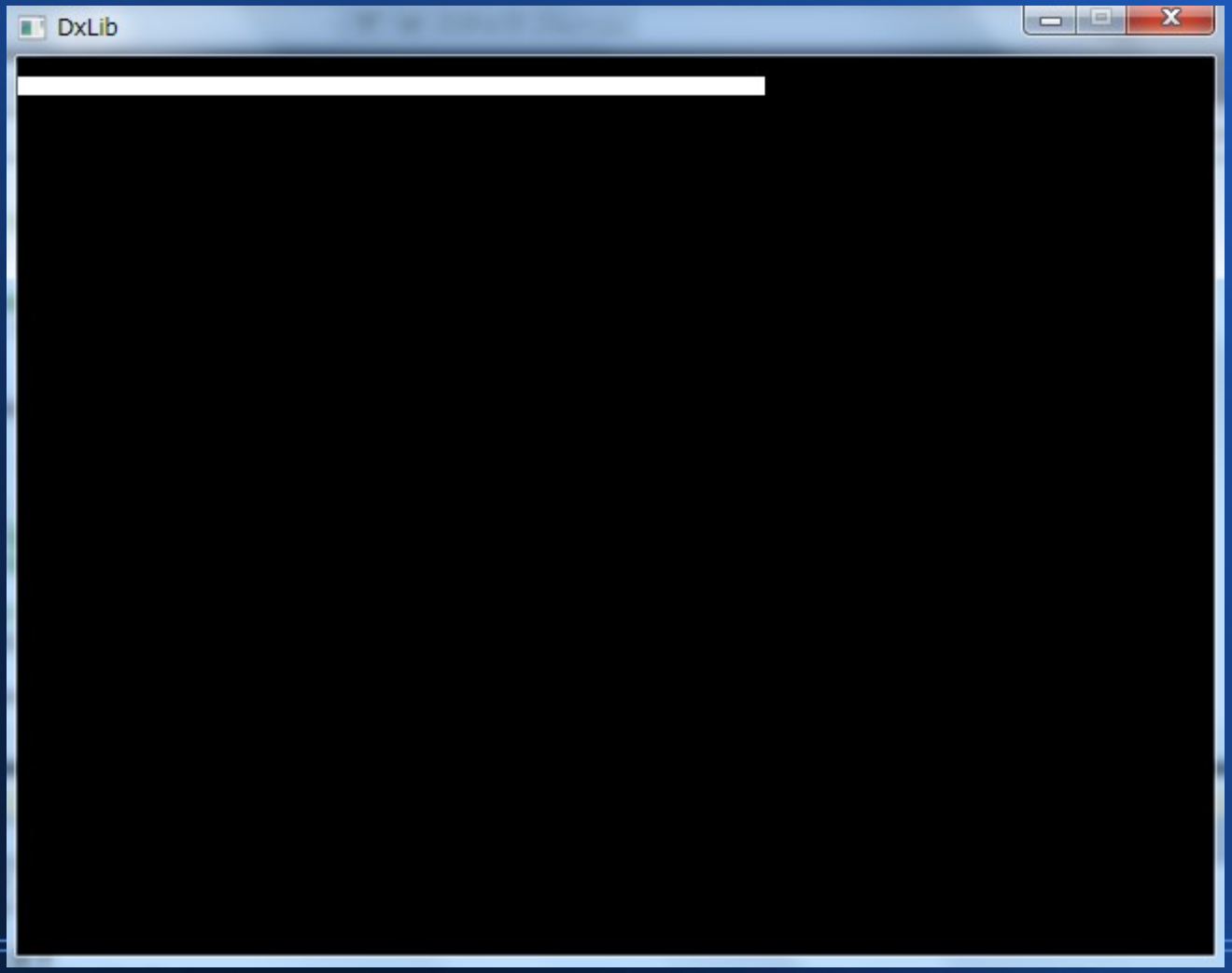
ソース例

```
#include "DxLib.h"

//x方向のブロックの数
#define BLOCK_X_NUM 10

// 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 sizeX=40;//箱の横の長さ
    int sizeY=10;//箱の縦の長さ

    //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！！
        //箱の描画
        for(int i=0;i<BLOCK_X_NUM;i++){
            int x1 = sizeX*i;
            int y1 = sizeY;
            int x2 = x1+sizeX;
            int y2 = y1+sizeY;
            DrawBox(x1,y1,x2,y2,GetColor(255,255,255),TRUE);
        }
```

箱の横の長さと
縦の長さを設定してやり、
x2はx1+sizeX
y2はy1+sizeY
でできる。

またx1をiに
比例するように作ると
次の箱の描画場所が
指定出来ることが
わかるだろう。

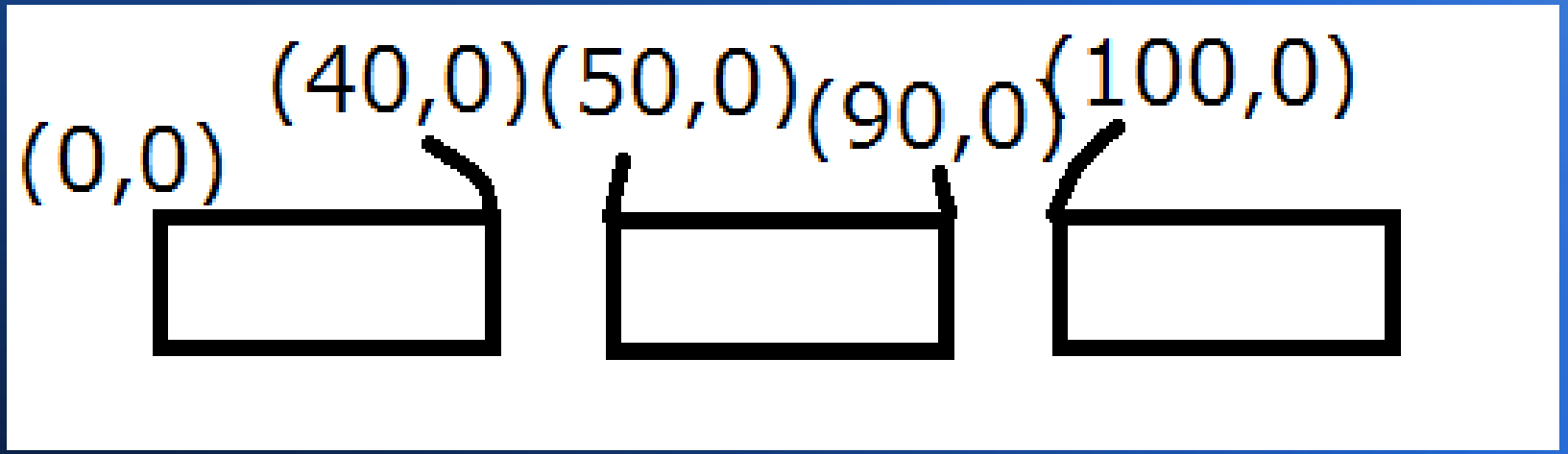
実行結果



これでも一応ブロックが
10個作られているのだが、
これだとよくないことは
わかるだろう。

どうしてこうなるかというと
1つめのx1は0,x2は40
2つめのx1は40,x2は80
とつながってしまう計算を
してしまったからである。

改善例



- イメージとしては、左上を50に比例するように設定。
- 右上を左上のx座標+40(sizeX)としてやればいい。

改善例

```
//メインループの動作はここに書く！！
//箱の描画
for(int i=0;i<BLOCK_X_NUM;i++){
    int x1 = (sizeX+10)*i;
    int y1 = sizeY;
    int x2 = x1+sizeX;
    int y2 = y1+sizeY;
    DrawBox(x1,y1,x2,y2,GetColor(255,255,255),TRUE);
}
```

（今回sizeXは40,sizeYは10である。）
こうすることにより、1つめのx1は50,x2は90
2つめのx1は100,x2は140となり、
前のx2と次のx1が重ならなくなることがわかる。

実行結果

このようにブロックが離れて
10個表示されたことが
わかるだろう。

#define BLOCK_X_NUMの
値を変えてやれば
ブロックの数も容易に増やせる

さらに縦にも
ブロックを増やしてみる。

- やることはx方向に増やした時とほぼ同じ。
- ただループが二重ループとなる。

- #define BLOCK_Y_NUM 5を追加して
  簡単に数を変更できるようにもしておく。

ソース例

```
//x方向のブロックの数
#define BLOCK_X_NUM 10
//y方向のブロックの数
#define BLOCK_Y_NUM 5

// 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 sizeX=40;//箱の横の長さ
    int sizeY=10;//箱の縦の長さ

    //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！！
        //箱の描画
        for(int i=0;i<BLOCK_X_NUM;i++){
            for(int j=0;j<BLOCK_Y_NUM;j++){
                int x1 = (sizeX+10)*i;
                int y1 = (sizeY+10)*j;
                int x2 = x1+sizeX;
                int y2 = y1+sizeY;
                DrawBox(x1,y1,x2,y2,GetColor(255,255,255),TRUE);
            }
        }
```

Y方向も
くっついてしまうのを
防ぐため
y1=(sizeY+10)*jとした
これにより、
前のブロックのy2と
次のブロックのy1が
重ならないようになる。

実行結果

このようにブロックがうまく作れたことがわかる。

GetColor関数の引数もiとjに比例するように作るときれいに色付けされるかもしれない。

まとめ

- 箱を描画する時はDrawBox関数
- ループを用いることにより、
  大量の図形を描画することが可能。
- 位置の設定は少し気をつけたほうがよい。