Windows 用簡便グラフィックスライブラリ eGGs の開発と評価*

髙比良 秀彰**

Development and evaluation of eGGs, an easy graphics library for Windows

Hideaki TAKAHIRA

１．はじめに

１．１．　開発の背景

現在，パーソナルコンピュータ（以下，PC）の OS としては，Windows OS（Microsoft 社。以下 Windows）が多く使われている。必然的に教育用や研究用の PC でも Windows を使うことが多くなっている。これはプログラミング教育や研究用プログラム開発においても同様である。

また，プログラミング教育や研究用プログラミングではコンソールプログラムが使われることが多い。プログラミング教育でコンソールプログラムを用いるのは，グラフィカルユーザインターフェース（以下 GUI）のプログラムに比べてプログラミングが簡単であり，プログラミングの本質を理解しやすいからである。また，研究用プログラムの開発においては，GUI のために開発時間を割くのが非効率であるからである。

一方でプログラミング教育や研究用プログラムにおいてもグラフィックスを利用した方がよい面もある。プログラミング教育に用いる言語は，入出力がテキストに限定されるものが多いが，筆者の経験ではグラフィックスを使った方がプログラミングに対する興味関心を維持しやすい。また，研究分野においては，たとえば画像処理研究ではグラフィックスの利用は必須ともいえる。

通常，プログラムからグラフィックス機能を利用する際には，グラフィックスライブラリ（以下 G ライブラリ）というものを使用する。ところが Windows にはコンソールプログラムから簡単に使用できる G ライブラリが標準では存在しない。

そこで筆者らが 1996 年から開発を続けてきた Windows 用 GUI ライブラリが GWC であり，eGGs の前身である。

現在では Windows で使用できる G ライブラリは，GrWin[1]や OpenCV[2]などが存在するが，当時，一般に公開されている G ライブラリはなく，自主開発するしかなかったことが開発を始めた理由である。なお，GWC は GUI や 2 次元グラフィックスのためのライブラリである。

１．２．　課題

１．２．１．　既存 G ライブラリに関する課題

現在よく名を聞くグラフィックスライブラリとしては，Cairo[3]，Direct2D[4]，OpenGL[5]などがある。また画像処理分野でよく使われるライブラリとして，グラフィックス表示と画像処理機能を融合させた OpenCV がある。

これらは背景に強力な開発体制をもち，高性能，高機能かつ利用のための様々な情報がインターネット経由で得られるライブラリであるが，実際にこれらを利用しようとしたことがあるものならばわかるとおり，大規模で，開発環境の構築にも手間がかかるものである。また Direct2D のように使い方そのものが難解なものもある。

すなわち，プログラム初学者がプログラミングを勉強するときに使ったり，研究分野で「サッと」グラフィックスを使いたいという要求に応えたりするようなライブラリになっていないさらには，一部の G ライブラリを用いて作成したプログラムは，そのライブラリの開発環境を構築した PC 以外で実行させることが難しい。

つまり似たような G ライブラリはあるものの，

① 使えるようにするための手間が小さく，
② 作成したプログラムを簡単に持ち運べ，

＊　原稿受付　令和元年 10 月 31 日

＊＊ 佐世保工業高等専門学校　電気電子工学科

③ 初心者にも簡単に使える

ような扱いやすいGライブラリは存在しない。

１．２．２．　ライブラリ開発の技術的課題

本稿で開発するのは，コンソールアプリケーションから使用する G ライブラリである。このようなライブラリの実行方式としては,コンソールの画面そのものをグラフィックスの描画に使用するタイプとコンソールとは別にグラフィックス専用の画面を持つタイプとに分類される。本稿で開発する G ライブラリでは後者の別画面を持つタイプを採用している。その理由はコンソールそのものを Character based User Interface（以下CUI）ツールとして用いることができる,複数のグラフィックス画面を同時に使用できるというメリットがあるためである。

しかしここで問題が生じる。それはグラフィックスの描画には時間がかかることに起因する。グラフィックスを描画している最中には,ライブラリを使用する本体のプログラムは計算などの処理を進めることができないという問題である。

そこで我々は,Gライブラリの仕様として，ライブラリを使用する際には呼び出し元,つまり本体のプログラムとは別にスレッド（別プログラム）を作成し，このスレッド上で描画処理を行わせることにした。これにより，ライブラリを使用するプログラムは G ライブラリが描画を完了するのを待たずに処理を続けることができるようになり,時間的ロスを大幅に削減できる。

ところがこの別スレッド方式によって別の問題が生じる。それはスレッドが独立したプログラムであるため，本体であるプログラムが，グラフィックス描画が完了しているかどうかを知ることができないということである。ライブラリは描画処理を行っている最中は別の描画命令を受け付けない。そのため，例えば何種類かの描画指令を連続して行うような場合にいくつかの描画が行われない可能性が高くなる。本稿ではこれを「描画落ち」と呼ぶことにする。

１．３．　開発の目的

そこで筆者は漠然と開発を続けてきた GWC において，前述の課題をクリアしたライブラリ eGGs を開発することとした。eGGs の開発ポリシーを以下に示す。

- プログラミング言語 C で開発すること
- 開発環境の構築が簡単であること。
- 作成したプログラムの運用が簡単であること
- ライブラリがコンパクトであること
- コンソールプログラムから使用できること
- 画像処理にも利用できること
- プログラムから描画中か否か確認できること
- 初心者でも直感的に使えること
- 手早くグラフィックスを描画できること

１．４．　本稿の概要

本稿では，まず２章で eGGs の設計およびプログラムとスレッドの連携手法について説明する。次に３章でライブラリの実装についてまとめる。さらに４章ではサンプルプログラム等により,本ライブラリが有用なグラフィックスライブラリであることを確認する。

さらに，eGGs のグラフィックスライブラリとしてのスタンスを明確にするために,画像処理およびグラフィックスライブラリのデファクトスタンダードである OpenCV との比較を行う。

さいごに各検証結果をまとめ，eGGs ライブラリの評価について述べる。

２．ライブラリの設計

２．１．　必要な機能

ここでは eGGs が装備すべき機能をあげるが，すべてを載せるには紙幅が不足するので抜粋して箇条書きで示す。

- コンソールアプリケーションで使用できること
- グラフィックス表示機能をコンソールと別に持つこと
- グラフィックス表示機能は，独立したウィンドウ（グラフィックウィンドウ。以下 GW）を作成したうえで，そのウィンドウ内で動作すること
- １つのアプリケーションで複数の GW を持てること
- GW の表示位置を指定できること
- GW のウィンドウタイトルを設定できること
- グラフィックデータを GW とは別に，メモリー上に構成できること（本データを以下 IGP と略す）
- IGP は複数保持できること
- GW および IGP に対して，基本的な幾何学図形（直線,

方形，円，楕円）および点を，座標の指定により描画できること
・図形等を描画する際に，既存の描画と重なった場合の描画方法を上書き，排他的論理和から選択できること
・GW および IGP に対して，文字を描画できること。また，文字は日本語も使用できること
・IGP の内容を GW に，座標を指定して転写できること
・GW および IGP 上の Pixel の色情報を座標指定により取得できること
・GW および IGP の一部を矩形で指定してコピーできること
・GW 上におけるマウスボタンクリックなどの操作情報を取得できること
・GW 上でのマウスカーソル位置情報を取得できること
・WindowsDIB 形式，JPEG 形式，PNG 形式の画像ファイルを読み書きできること
・GW に対するキー入力を取得できること
・アプリケーションからの命令により，GW を閉じて破棄できること
・GW に描画されている内容を，DIB，JPEG，PNG の各形式でファイルに保存できること

２．２．　満たすべき性能

eGGs が満たすべき性能のうち主要なものを列挙する。

・GW におけるグラフィックス表示遅延が，プログラムの母体となるコンソールアプリケーションの動作に影響を与えないこと
・マウスによるお絵かきなどのようなリアルタイム描画や，描画切り替えによるスムーズなアニメーションが実現できる描画速度であること
・１ヘッダ，１実行ファイル，１ライブラリ構成であること
・C 言語で開発し，他の言語からも利用可能であること

２．３．　付加機能

２．１の必要な機能に加えて，直感的使用を助けるため，また，様々なアプリケーションを開発できるようにするなどのために付加した特徴的機能を列挙する。

・C の printf 関数と同等機能を持つ文字列表示
・プログラム内の命令による GW の移動
・マスク描画機能
・GW の拡大縮小表示
・GW 拡大縮小表示時のマウス座標取得
・GW 描画領域のスクロール
・GW 外のマウス座標取得
・デスクトップの大きさを得る
・文字表示領域計算
・エディットコントロール機能
・ボタンコントロール機能
・ウィンドウタイトル変更
・文字の描画時の Windows フォント指定
・WindowsDIB データ形式のオンメモリー使用
・メモリー上の DIB データの複製
・ウィンドウフォーカスの切り替え
・GW の形を自由形状にする
・GW に描画する際の更新領域を限定する
・GW の描画内容をクリップボードにコピーする

２．４．　実装設計

上述の機能，性能を実現するために考慮すべき実装上の留意点について列挙する。

・動的リンクライブラリとする
・GW ごとにスレッドをわける
・GW は１個につき１つのハンドル変数で管理する
・背景色や，線の種類，塗りの種類，マウス情報などはすべて GW ごとに独立とする
・類似機能における機能の切り替えは，引数による指定とせず，独立した関数にする
・DLL とアプリケーションの協調機能を持たせる
・高速化のためのシンプルな関数を用意する

２．５．　本体プログラムとライブラリ連携

１章の１．２．２で述べたとおり，ライブラリを使用するプログラムとライブラリスレッドには連携が必要である。eGGs ではこの連携のために，描画状態を示すフラグを使用する。

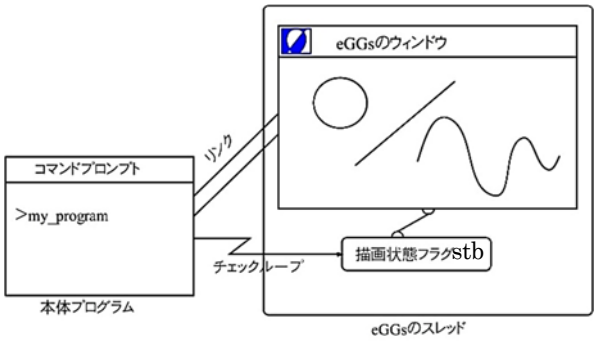


図１　描画状態を示すフラグ

図１は,本体プログラムと eGGs の連携方式を示す。スレッドには１つのグラフィック描画用ウィンドウと描画状態を表す変数，描画状態フラグ（stb）を持たせる。stb は２つの状態 TRUE と FALSE のいずれかを持ち,TRUE は描画処理がすべて完了し次の描画処理が可能であることを意味し，FALSE は描画中で次の描画ができないことを意味する。スレッドが stb の書き換えを行い，本体プログラムは stb の値の参照のみを行うことができる。

動作は次の通りである。本体プログラムにより描画の指示が eGGs のスレッドに送られると，スレッドにより stb が FALSE にセットされる。そして描画が完了すると stb は TRUE にセットされる。連続して描画を行う際には，本体プログラムは stb を繰り返し参照（チェックループ）し，stb が TRUE になったら次の描画指示を送る。この仕組みにより，eGGs は描画落ちをすることなく連続描画を実行できる。

３．ライブラリの実装

２の設計に基づいて，Visual Studio 2019 の C 言語により eGGs を作成した。eGGs とは enhanced Graphics Gear system の略称である。eGGs を構成するファイルは,

- libgg.dll…ライブラリ本体
- libgg.h　…ヘッダファイル
- libgg.lib…インポートライブラリ

の３ファイルとした。なお，libgg.lib は Visual Studio の C あるいは C++コンパイラ（cl.exe）用インポートライブラリであり,他のコンパイラで使用する場合は，別途インポートライブラリを生成するか，動的リンクを行う必要がある。

eGGs は C 言語で開発されており，本ライブラリは C 言語で使用するのが望ましいが，ライブラリ本体が C コンパイラにより作成された DLL 形式で提供されることから多言語での使用もたやすい。

４．　ライブラリの評価

ここでは,まず eGGs が G ライブラリとして正常に使用できるかどうかを検証し,次に各課題がクリアされているかどうかを確認する。さらに，OpenCV と比較することによって，どのような場面で eGGs が使用できるかを明確にする。

４．１．　ライブラリの動作検証

eGGs の動作検証として，まず GW 作成，図形描画など，２．の設計にあげた各項について個別にテストプログラムを作成し,すべての機能が正常に動作することを確認した。テストの数は膨大であるため本稿では割愛するが,これによりライブラリは仕様のとおり正しく開発されたことが確認できた。

次に，eGGs が画像処理に使用できるかどうかを検証した。画像処理プログラミングにおいては，原則として次の５つの機能が必要となる。すなわち，これらが使えれば,画像処理および処理結果のグラフィックス表示を行う G ライブラリとしての最低条件はクリアできている。

（１）　画像ファイルの読み込みができる
（２）　画素情報の読み出しができる
（３）　画素情報の書き換えができる
（４）　画像の表示ができる
（５）　画像の書き出しができる

検証のため eGGs を用いて赤画素抽出プログラムを作成した。リスト１がそのコードである。なお，本稿ではプログラムを作るためのライブラリ開発とその評価が目的のため,ライブラリ検証用に必要最小限でプログラムソースを載せている。

リスト１の処理は原画像中の赤色の画素のみを抽出し，結果の画像に黒で示すものである。図２，図３より，本処理が正常に行われていることがわかる。この処理では，画像の読み込み = gg_loadJpeg，画素情

報の読み出し = gg_getDibPixel，画素情報の書き換え = gg_putDibPixel，画像の表示 = gg_displayDib，画像の保存 = gg_savePngFrmDib のすべての処理が行われており，eGGs が画像処理プログラミングに使用できることを示している。

リスト１　サンプルプログラム

```
int main( void )
{
    pGC g1 , g2;
    pDIB src,dst;

    src = gg_loadJpeg( "input-768x576.jpg" );
    GGSTART( g1, "原画像", w, h , 10 , 10 );
    GGSTART( g2, "赤画像", w, h , 10+w+10 , 10 );

    gg_displayDib( g1 , 0,0 , src );
    dst = get_fbmp( src );
    gg_displayDib( g2 , 0,0 , dst );
    gg_savePngFrmDib( dst , "newimg.png" );
}

pDIB get_fbmp( pDIB img )
{
     pDIB dst = gg_copyDibitmap( img );
     for ( y=0 ; y<h ; y++ ) {
         for ( x=0 ; x<w ; x++ ) {
             c = gg_getDibPixel( img , x , y );
             if ( isRED(c) )
                 gg_putDibPixel(dst,x,y,BLACK );
             else
                 gg_putDibPixel(dst,x,y,WHITE );
     }
     return dst;
}
```

図２　原画像

図３　赤画素抽出結果

４．２．　インストール容易性の評価

eGGs のインストールは，３つのファイルをプログラムソースファイルと同じディレクトリにコピーするだけである。

これは,OpenCV をはじめとする他の G ライブラリがインストーラを用いてインストールされ,通常はインストール先のディレクトリ構成,ファイル配置も隠蔽されているのに比べると,簡単かつ明解であることは自明である。

４．３．　作成したプログラムの可搬性の評価

作成したプログラムを別の PC で簡単に使えるか否かが可搬性の評価である。

eGGs においては，プログラムファイルと同じディレクトリに libgg.dll があれば実行可能である。したがって，開発をおこなった PC と異なる PC でプログラムを使用したい場合は，プログラムファイルと libgg.dll ファイルのみを同じフォルダにコピーするだけでよい。

他の多くのグラフィックスライブラリでは，一般的にプログラムを使用したい PC にライブラリをインストールする必要があることから，eGGs を用いたプログラムの可搬性の高さは明白である。

また，可搬性の評価としてはライブラリ本体のサイズもポイントとなる。ライブラリのファイルサイズが大きければ，運用先の PC のストレージや運搬に使用するポータブルストレージの容量を圧迫するからである。eGGs の本体である libgg.dll ファイルのサイズは 1.02M バイトしかないため，可搬性は高いといえる。

４．４．　OpenCV との機能比較

画像処理プログラミングのデファクトスタンダードともいえるオープンソースの画像処理ライブラリ OpenCV は画像処理のみならず，グラフィックインターフェースライブラリとしてもしばしば使用されている。本稿では，我々が開発した eGGs の機能とスタンスを明確にするため,主要な機能および性能についてOpenCV と比較を行い,その結果を表1にまとめた。

この表からわかるとおり,画像表示インターフェースとしての機能差はほとんどなく，OpenCV では多種多様な画像処理アルゴリズムが実装されているなど画像処理ライブラリとしての性格が強いのに対して，eGGs では WindowsAPI が使用できる，日本語表示ができるなど GUI としての性格が強いといえる。

表１　eGGs と OpenCV の機能比較

機能	eGGs	OpenCV
表示用ウィンドウの作成	○	○
画像ファイルの読み書き	○	○
画素情報の読み書き	○	○
内部画像データの生成	○	○
画像データの表示	○	○
基本的図形の描画	○	○
文字の表示	○	△
マスク処理	○	○
ウィンドウの移動	○	○
ウィンドウの拡大縮小	○	○
マウス情報取得	○	○
キー入力	○	○
スライダーコントロール	×	○
ボタンコントロール	○	○
エディットボックスコントロール	○	×
画像関連 WindowsAPI の直接利用	○	×
画像処理アルゴリズム実装	×	○
マルチプラットフォーム	×	○

○：実装されている
×：実装されていない
△：実装されているが制限がある

４．５．　機能使用の容易さと可読性の評価

ここでは eGGs と OpenCV のサンプルコードを提示し,ライブラリの各機能を使用するコードのわかりやすさを評価する。コードがわかりやすいと言うことは,ライブラリ機能の使用が容易であること,可読性が高いことにつながるからである。

４．５．１．　GW の作成

まず GW を１つ作成する eGGs のコードを示す。

```
GGSTART(gw, "ウィンドウ名", 400 , 300 , 0,0 );
```

“GGSTART”は GW を１つ作成する命令であり，gw は GW を管理するための変数である。“400,300”は GW の幅と高さ，“0,0”はデスクトップ上の GW の表示位置である。“ウィンドウ名”は GW のタイトルバーのタイトルである。

次に OpenCV を用いてウィンドウを１つ作成するコードを示す。

```
namedWindow( "gw" , CV_WINDOW_AUTOSIZE );
resizeWindow( "gw" , 400 , 300 );
moveWindow( "gw" , 0 , 0 );
setWindowTitle( "gw", "ウィンドウ名" );
```

２つのコードを比較すればわかるが，OpenCV によるコードの方が煩雑なものとなっている。

また,ここで注意すべきはウィンドウの指定方法である。eGGs ではウィンドウを変数で管理するのに対して，OpenCV ではウィンドウに名前をつけて管理するため，コードの保守が煩雑になる可能性がある。

次に GW を２つ作成するコードと実行結果を示す。

```
GGSTART(gw1, "１つめ",400,300,0,0);
GGSTART(gw2, "２つめ",800,400,410,0);
```

OpenCV によるコードは，上の OpenCV のコードの繰り返しとなるため割愛するが，eGGs の方がシンプルでわかりやすいといえる。

４．５．２．　基本的な図形の描画

ここでは,基本的な図形の一例として直線を描画するコードを比較する。まずは eGGs によるコードである。

```
GGSTART( gw , "直線描画" , 400 , 300 , 0,0 );
gg_drawLine( gw , 10 , 10 , 200 , 200 , RGB(0,0,0) );
```

ここで，gg_drawLine が直線を描く API である。gw でウィンドウを指定し,座標(10,10)から(200,200)へ直線を引くよう指定してある。また，RGB(0,0,0)は線の色の指定である。括弧の中はコンマで３つに区切られており，先頭から赤成分，緑成分，青成分を示している。

同じ処理を行うコードを OpenCV で書いたものを以下に示す。

```
namedWindow( "gw" , CV_WINDOW_AUTOSIZE );
moveWindow( "gw" , 0 , 0 );
setWindowTitle( "gw" , "直線描画" );
Mat image(300,400,CV_8UC3 , Scalar( 255,255,255) );
Point p1(100,100), p2(200,100);
Scalar colorLine(0,0,0);
line(image, p1, p2, colorLine, 1);
imshow( "gw", image );
```

上から 3 行は GW を作成するためのコードなので，直線を描くコードは 4 行目以降である。OpenCV は基本的に画像を表示するライブラリであるため,プログラム内で画像データを生成し，これに直線を描く。その後,直線を描いた画像をウィンドウに表示するため,コードの分量が多くなっている。

４．５．３．　文字列の表示

最後に文字列を表示するコードを示す。座標（20,100）に”Hello”という文字列を赤色で表示するコードである。まずは，eGGs の場合である。

```
GGSTART( gw , "文字列描画" , 400 , 300 , 0 , 0 );
gg_putText( gw, 20,100 , "Hello" , RGB(255,0,0) );
```

次に OpenCV のコードを示す。

```
namedWindow( "gw" , CV_WINDOW_AUTOSIZE );
moveWindow( "gw" , 0 , 0 );
setWindowTitle( "gw" , "文字列描画" );
Mat image(300,400,CV_8UC3, Scalar(255,255,255));
putText(image,"Hello",Point(20,100),
        FONT_HERSHEY_SIMPLEX, 2, Scalar(0,0,255),
        1, CV_AA);
imshow( "gw", image );
```

これも，４．５．２の場合と同様，4 行目からが文字列を表示するコードである。

文字列の表示では，eGGs，OpenCV ともに若干の制限がある。まず，eGGs における制限である。ここに示した最もシンプルな文字列表示 API では，フォントが Windows の標準フォントに限定され，フォントの種類，大きさ，太さは変更できない。フォントを変更する場合には，専用の API を使用しなければならず，やや煩雑である。次に OpenCV の制限である。OpenCV では基本的に Windows にインストールされている様々なフォントを使用することができず,OpenCV 独自のフォントとなる。ただし，大きさ，線の太さ，色が変更可能である。なお，OpenCV では日本語には対応しておらず，日本語の表示はできない。

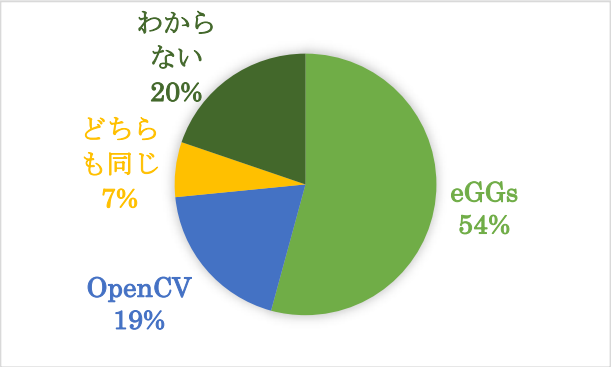
本稿でのコード記載による比較は以上 3 例とし，他の例は割愛するが,他の機能を使用するコードにおいても全般的に eGGs の方が簡潔で見やすいコードとなっている。

４．５．４．　アンケートによる評価

以上のように，eGGs のコードは OpenCV によるコードに比べて短く,可読性が高いものとなっているのは自明と思われるが,客観的な評価のためプログラミング初学者 43 名を対象にアンケート評価も行った。アンケートでは,１)グラフィックスウィンドウの表示,２）直線の表示，３）文字の表示，４）画像ファイルの読み込みと表示の各処理について eGGs によるコードと，OpenCV によるコードを提示し,

- **eGGs コードがわかりやすい**
- **OpenCV コードがわかりやすい**
- **どちらも同じくらいのわかりやすさ**
- **わからない（判断できない）**

以上の 4 項目から１つを選択してもらうというものである。グラフ１にアンケート結果を示す。この結果は前述１）から４）の各処理についての回答の平均をとったものである。



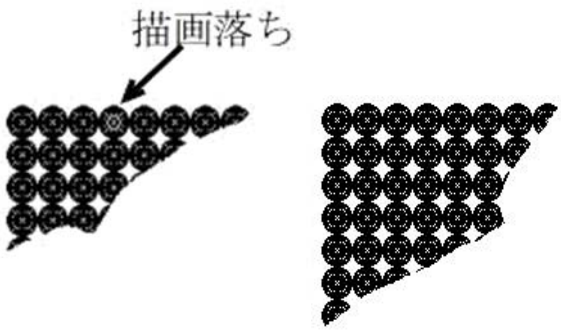
グラフ 1　eGGs と OpenCV のコード可読性の比較

これからわかるとおり，やはり eGGs のコードの方が OpenCV のコードよりわかりやすいという意見が多かった。

４．６．　連続描画時の描画落ちの検証

以下に示す，中心座標，半径を変化させながら 4000 個の円を描くコードにより，描画落ちが発生しないか検証を行った。

```
for ( i=1 ; i<=20 ; i++ )
  for ( j=1 ; j<=20 ; j++ )
    for ( r=1 ; r<=10 ; r++ ) {
      gg_drawCircle(gw, j*20, i*20, r, RGB(0,0,0) );
      while (stb!=TRUE);…描画落ち対策コード
    }
```



(a)　　(b)

図 4　描画落ちの検証結果

図 4 はコードを実行した結果である。(a)は描画落ち対策コードを除いて実行したもの，(b)は描画落ち対策コードを入れて実行したものである。図からわかるとおり，stb を使った描画落ち対策がないコードでは(a)の矢印の位置に，周りの円形に比べて白い円が目立っていて描画落ちが発生している。対して(b)の対策を行ったコードでは描画落ちは発生していない。これより，描画状態フラグ stb を用いる eGGs と本体プログラムの連携はうまくいっていることがわかる。

５．おわりに

本稿では，筆者が開発したグラフィックスライブラリ eGGs が，インストールの難易度が低く，作成したプログラムの可搬性が高く，かつ，グラフィックスライブラリが抱える描画落ちの解決手法を提供するライブラリであることを示した。この描画落ち対策は，本体プログラムとは別にスレッドを生成して処理を行うような他のライブラリに対しても有効な手段であることは自明である。

また，このライブラリが画像処理プログラミングに使用できることを示し，さらに画像処理プログラミングおよび GUI 用のライブラリである OpenCV と比較をすることによって，高度な画像処理機能を要求しないならば画像処理プログラミングにおいても有用なライブラリであることを示した。さらに，eGGs と OpenCV の実際のプログラムコードを比較することでプログラム初学者にとっても eGGs はわかりやすいライブラリであることを示した。

結論として，eGGs はグラフィックス表示を主目的とする GUI ツールとして使ったり，初歩的画像処理ツールとして使用したりするならば，有用なライブラリであるといえる。

今後の予定としては，まずは他のライブラリが持つ動画処理機能を実装してみたい。

参考文献

1) Tsuguhiro TAMARIBUCHI , GrWin, https://spdg1.sci.shizuoka.ac.jp/grwin/ja/,last update 2019.2.7;2019.10.31 確認
2) OpenCV team, OpenCV 公式サイト，https://opencv.org/，2019;2019.10.31 確認
3) Cairo 公式サイト，https://www.cairographics.org/，last update 2016.4.25,2019.10.31 確認
4) Microsoft, Direct2D, https://msdn.microsoft.com/ja-jp/windows/desktop/dd370990，2019;2019.10.31 確認
5) Khronos Group, OpenGL 公式サイト，https://www.opengl.org/，2019;2019.10.31 確認