drawlib

rev. 2.2
2021 年 4 月 15 日
藤江 真也

1. はじめに

drawlib（ドローリブ）は，C 言語のプログラミングで手軽にお絵かきをしたり，キーボードやマウスの操作ができるように意図されて開発されたライブラリです．

2. 導入

2.1. OpenCV のインストール

下記を実行して，OpenCV をインストールする．

```
$ sudo apt install libopencv-dev
```

2.2. drawlib の取得とインストール

Ubuntu 18.04 LTS を使っている人は，以下を実行してください．

```
$ curl -JLO https://bit.ly/drawlib102u18
$ sudo tar zxvf drawlib-1.0.2-ubuntu_18.04_64bit.tgz -C /
$ sudo ldconfig
```

2.3. サンプルプログラムの作成と実行

下記のプログラムを `test.c` というファイル名で作成する．

プログラムに含まれる文字の色は，見やすさのためにつけているので無視して良い．

```c
#include <drawlib.h>

int main(void) {
  dl_initialize(1.0);

  dl_line(10, 10, 100, 100, dl_color_from_name("blue"), 2);

  while (1) {
    dl_wait(0.01);
  }

  return 0;
}
```

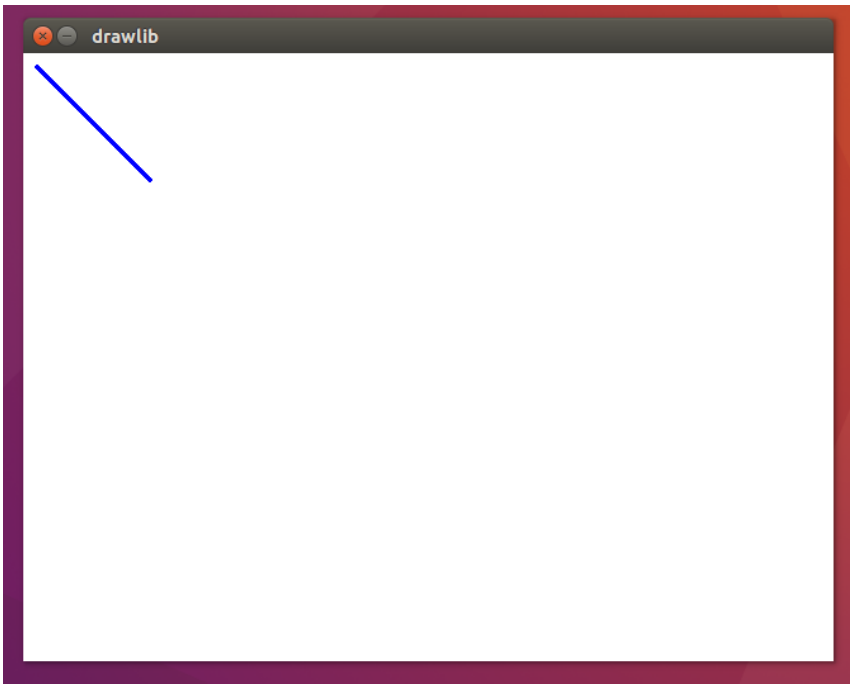
下記のようにコンパイル＆リンクを行う．

```
$ gcc -o test test.c -ldrawlib
```

とくにエラーが出なければ実行する．

```
$ ./test
```

実行すると，以下のようなウィンドウが表示される．
ターミナル（端末）のウィンドウの裏に出ている場合があるので注意する．



プログラムの停止は，ターミナル上で Ctrl-C（ctrl キーを押しながら c キーを押す）で行う．

3. drawlib を使ったプログラミング

3.1. サンプルプログラムの説明

サンプルプログラムは実行結果を見ての通り，青い線が斜めに引かれた線を表示する簡単なものである．ここでは，test.c の内容を見ながらどのようなプログラムでこれが実現できるか説明する．

まず，1 行目の

```
#include <drawlib.h>
```

はヘッダを読み込む部分で drawlib ライブラリを使うために必ず必要である．

次に，main 関数内冒頭の

```
  dl_initialize(1.0);
```

では，drawlib ライブラリの初期化を行い，ウィンドウが表示される．なお，drawlib の関数にはすべて dl_ という接頭辞がついている．引数の 1.0 は表示されるウィンドウのスケールを表し，実数で与えられる．例えばこれを 2.0 にすると，横幅，縦幅ともに 2 倍される．

```
  dl_line(10, 10, 100, 100, dl_color_from_name("blue"), 2);
```

では，ウィンドウ上に線が引かれる．dl_line 関数はウィンドウに線を引く関数である．引数に与えられる値によって線をどのように引くかが決まる．

- 最初の 4 つの引数，10, 10, 100, 100 は，線の始点と終点の座標を表す．ここでは点(10, 10)から点(100, 100)に線が引かれることになる．
- 5 番目の引数には色を指定する．ここでは dl_color_from_name という別の関数の呼び出しの結果を与えている．dl_color_from_name は引数として色の名前を指定することで，その色に対応する値を取得することができる．
- 最後の引数に与えられている 2 は，線の太さを表している．

この後，プログラムがすぐ終了しないように while(1)で永久ループしている．ループの中の

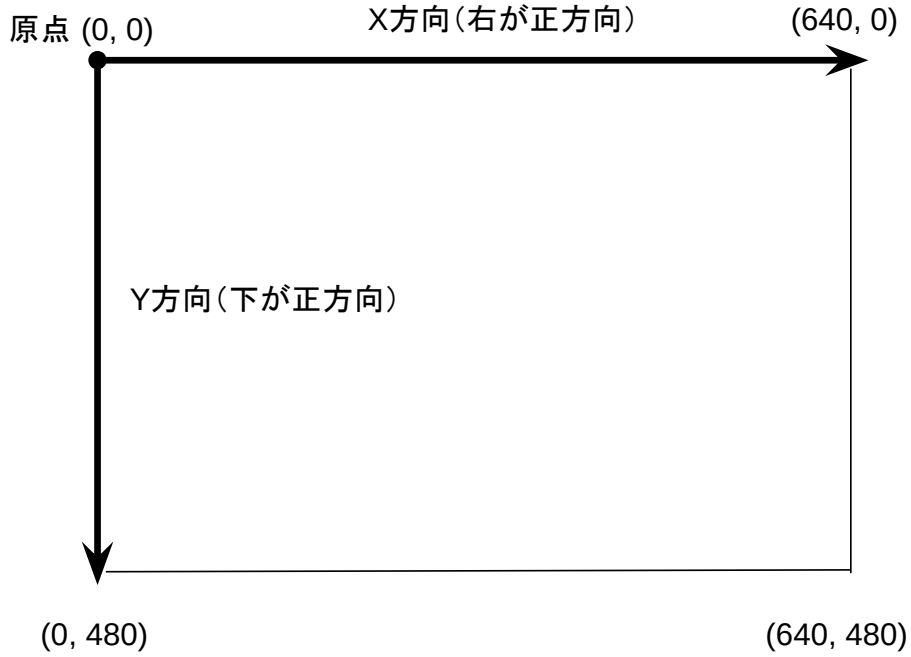
```
    dl_wait(0.01);
```

は，プログラムの実行を待機する関数で，引数により待機する時間を指定できる．引数は単位が秒の実数で指定する．ここでは 0.01 秒待機することになる．この部分がないとプログラムの実行が無駄に重くなり，コンピュータに負荷をかけることになることがあるため注意が必要である．

3.2. ウィンドウ内の座標

drawlib では，dl_initialize で表示されるウィンドウに，線などの図形や文字の描画を行う．このとき，ウィンドウのどの位置に描画を行うかは，2 次元平面上の点を

X 座標と Y 座標で与えることで指定する．ウィンドウ内の座標は次の図のようになっている．



原点は左上にあり，その座標は(0, 0)である．点の座標は，X 座標の値を x，Y 座標の値を y としたときに(x, y)と表す．X 方向は右に向かって正で，Y 方向は下に向かって正である．特に Y 方向は数学で使う座標軸の方向と逆なので注意する．

ウィンドウ内に描画された図形や文字は，複数のピクセル（またはドット）と呼ばれる小さな領域に色をつけることで表されている．ピクセルはウィンドウ内で縦横に隙間なく並べられており，drawlib のウィンドウでは横に 640 個，縦に 480 個ならんでいる．

各種図形や文字を描画する際には，このピクセルの位置を点の座標として与えることになる．その時の値の範囲は，x 座標が 0〜639，y 座標が 0〜479 となる．ピクセルの大きさは縦横の幅がそれぞれ 1 と決まっているため，整数で与える（1.0 未満の値は有効でない）．

3.3. 図形や文字の描画と色

3.3.1. 図形や文字の描画

図形の描画は専用の関数を用いて行う．ここでは，関数名とその機能の概略だけを述べる．それぞれの詳細は後述のリファレンスを参照すること．

- `dl_line` は，2 点を結ぶ線分を描画する．
- `dl_rectangle` は，2 点を頂点とする矩形（長方形または正方形）を描画する．塗りつぶすこともできる．

- dl_circle は，円を描画する．中心点の座標と半径を与える．塗りつぶすこともできる．
- dl_ellipse は，楕円，もしくは円弧を描画する．中心点の座標，長径と短径，傾ける角度，開始角度，終了角度を与える．塗りつぶすこともできる．
- dl_text は，文字列を描画する．

3.3.2. 色

描画する図形の色は，それぞれの描画関数の引数に long 型の整数値として与えるが，直接与えることは難しいので，サンプルプログラムにあるように dl_color_from_name 関数を使って名前から値を求めるか，dl_color_from_rgb 関数を使って RGB 値から求めるのが好ましい．

dl_color_from_name 関数に与えられる色の名前は以下の通りである．

- black ... 黒
- red ... 赤
- yellow ... 黄
- magenda ... マゼンタ（紫）
- green ... 緑
- cyan ... シアン（水色）
- blue ... 青
- white ... 白

上記の色の指定だけでは中間的な淡い色を出すことができない．dl_color_from_rgb 関数を使うことで，赤の輝度，青の輝度，緑の輝度を 0〜255 の強さで含む色の値を取得することができる．dl_color_from_rgb 関数についての詳細は後述のリファレンスを参照すること．

4. イベントの処理

4.1. イベント処理のサンプルプログラム

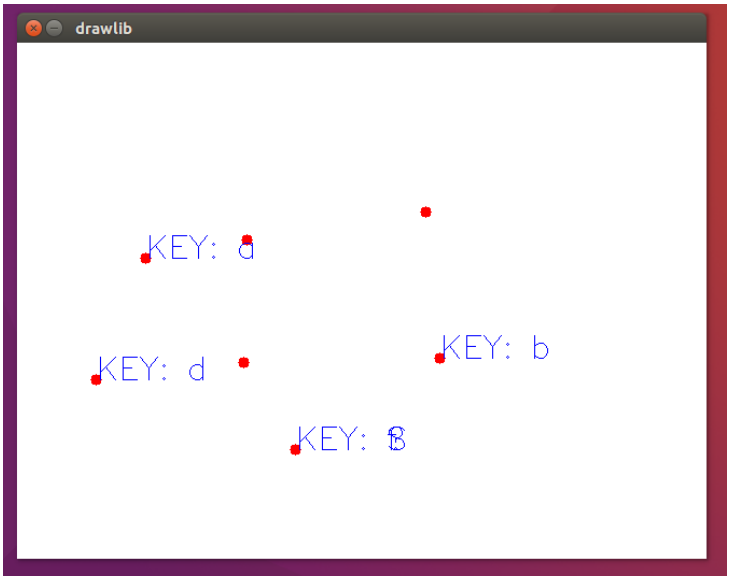
次のプログラムをコンパイルして実行してみる．

```c
#include <stdio.h>
#include <drawlib.h>

int main (void) {
  int t, k, x = 100, y = 100;
  char str[10];

  dl_initialize(1.0);
  while (1) {
    if (dl_get_event(&t, &k, &x, &y)) {
      if (t == DL_EVENT_KEY) {
        sprintf(str, "KEY: %c", k);
        dl_text(str, x, y, 1.0, dl_color_from_name("blue"), 1);
      } else {
        dl_circle(x, y, 5, dl_color_from_name("red"), 1, 1);
      }
    }
    dl_wait(0.01);
  }
  return 0;
}
```

このプログラムを実行すると，何も描かれないウィンドウが表示される．ウィンドウ上でマウスをクリックしたり，キーを押すことで以下のような動作をする．

- マウスで左クリックをすると，マウスカーソルの位置に赤い小さな円が描画される．
- キーボード上の任意のキーを押すと，最後に円を描画した位置に「KEY: ?」という文字列が青色で描画される．このとき?は押したキーに対応するキャラクタである．

例えば以下のような表示がされる．



4.2. サンプルプログラムの説明

drawlib のウィンドウの上で行われたマウス操作やキー操作についてはイベントとして蓄積される．蓄積されたイベントは dl_get_event 関数を使って取得することができる．

イベント処理のサンプルプログラムの無限ループ（while(1)）のブロック内での処理は，以下のような意味を持つ．

- `if (dl_get_event(&t, &k, &x, &y)) {`
  - dl_get_event 関数は，蓄積されたイベントを取得する．
  - イベントが取得できた場合は 1（真）を返し，取得できない場合は 0（偽）が戻り値となる．
  - イベントが取得できた場合は，t にイベントの種類，k に押されたキー．x，y にマウスが押下された x 座標，y 座標が代入される．
  - イベントは，蓄積されたものの中で最も古いものを取得し，消去される．蓄積されていない（すべてが消去されている）状態で呼ばれた場合は 0 が戻り値となる．
  - 上記のように if 文の条件として呼び出すことで，イベントが取得できた場合は続くブロック内を実行し，イベントが取得できなかったときは何もしない，という分岐が実現できる．
- `if (t == DL_EVENT_KEY) {`
  - イベントのタイプに応じて分岐させることを意図している．
  - t には dl_get_event 関数で取得したイベントの種類が入っている．イベントの種類は以下のうちのいずれかである．
    - DL_EVENT_KEY ... キーボード上のいずれかのキーが押された．
    - DL_EVENT_L_DOWN ... マウスの左ボタンが押された．
    - DL_EVENT_LUP ... マウスの左ボタンが離された．
    - DL_EVENT_R_DOWN ... マウスの右ボタンが押された．
    - DL_EVENT_R_UP ... マウスの右ボタンが離された．
  - 今回の場合は，キーボード上のいずれかのキーが押された場合に続くブロックが実行される．その他（マウスのボタンに関する操作）の場合は，else 節以降が実行される．
- `sprintf(str, "KEY: %c", k);`
  - 文字列（文字配列）に，押下されたキーに対応するキャラクタを含む値を設定する．sprintf 関数は stdio.h に用意されている標準ライブラリの関数である．
  - k には dl_get_event 関数で取得した，押されたキーに対応する数値が代入されている．基本的にはキーの ASCII コードに対応した数値が与えられるが，矢印キーなどの ASCII コードで表せないものは別の数

値が与えられる．詳細は後述の dl_get_event 関数のリファレンスを参照すること．

- `dl_text(str, x, y, 1.0, dl_color_from_name("blue"), 1);`
    - dl_text 関数を使って配列 str の内容を座標(x, y)に描画する．
    - 引数の詳細については後述の dl_text 関数のリファレンスを参照すること．
- `dl_circle(x, y, 5, dl_color_from_name("red"), 1, 1);`
    - else 節に書いてある文なので，イベントの種類（t）がキーの押されたもの（DL_EVENT_KEY）以外の場合，すなわちマウスのボタン操作の場合に実行される．
    - 中心点(x, y)，半径 5 の円を描画する．x，y には，dl_get_event 関数によってマウスのボタン操作が行われた位置の x 座標と y 座標がそれぞれ代入されているため，マウスのボタンが押された（あるいは離された）位置に円が描画される．その他の引数に関しては後述の dl_circle 関数のリファレンスを参照すること．

5. リファレンス

5.1. 初期化関数

5.1.1. int dl_initialize(double scale);

概要

drawlib を初期化し，ウィンドウを表示する．

引数

- scale … 表示されるウィンドウのサイズが scale 倍される．1.0 を指定すると幅が 640，高さ 480 のウィンドウが表示される．

戻り値

初期化に成功した場合は 0，失敗した場合は 1．

5.2. ユーティリティ関数

5.2.1. long dl_color_from_name(const char* name);

概要

名前に対応する色の数値を取得する．
DL_C(name)でも同様の動作をする．

引数

- name … 値を取得したい色の名前を指定する．有効な値は以下の通り．
  - black … 黒
  - red … 赤
  - yellow … 黄
  - magenda … マゼンタ（紫）
  - green … 緑
  - cyan … シアン（水色）
  - blue … 青
  - white … 白

戻り値

指定した色に対応する整数値．

注意

無効な名前が指定された場合はエラーメッセージが表示され，white を指定したときと同じ値が得られる（プログラムは中断しない）．

5.2.2. long dl_color_from_rgb(int r, int g, int b)

概要

R，G，B の強さ（輝度）に対応する色の数値を取得する．
DL_RGB(r, g, b) でも同様の動作をする．

引数

- r … R（赤）の輝度値．0〜255 で与える．
- g … G（緑）の輝度値．0〜255 で与える．
- b … B（青）の輝度値．0〜255 で与える．

戻り値

引数で与えた R，G，B の値に対応する整数値.

r，g，b に与える値と実際の色の例を以下に示す.

- 128, 128, 128
- 128, 0, 0
- 128, 128, 0
- 255, 128, 0
- 128, 255, 128

注意

各引数に範囲外の値が指定された場合の動作に関しては不定である（プログラムは中断しないが想定外の色が出力される）.

5.2.3. void dl_wait(float sec)

概要

プログラムの実行を指定された時間，待機する.

引数

- sec … 待機する時間を秒単位で指定する.

5.3. 描画関数

5.3.1. void dl_line(int x1, int y1, int x2, int y2, long color, int thickness)

概要

線分を描画する.

引数

- x1 … 始点の x 座標
- y1 … 始点の y 座標
- x2 … 終点の x 座標
- y2 … 終点の y 座標
- color … 色に対応する整数値（dl_color_from_name 関数を参照のこと）
- thickness … 線の太さ（1 が標準．大きいほど太い線が描画される）

5.3.2. void dl_rectangle(int x1, int y1, int x2, int y2, long color, int thickness, int fill)

概要

矩形（長方形，または正方形）を描画する．

引数

- x1 … 1 つ目の頂点の x 座標
- y1 … 1 つ目の頂点の y 座標
- x2 … 2 つ目の頂点の x 座標
- y2 … 2 つ目の頂点の y 座標
- color … 色に対応する整数値（dl_color_from_name 関数を参照のこと）
- thickness … 線の太さ（1 が標準．大きいほど太い線が描画される）
- fill … 1 を指定すると，塗りつぶされる．0 を指定すると塗りつぶされない．

5.3.3. void dl_circle(int x1, int y1, int radius, long color, int thickness, int fill)

概要

円を描画する．

引数

- x1 … 中心点の x 座標
- y1 … 中心点の y 座標
- radius … 半径の長さ
- color … 色に対応する整数値（dl_color_from_name 関数を参照のこと）
- thickness … 線の太さ（1 が標準．大きいほど太い線が描画される）
- fill … 1 を指定すると，塗りつぶされる．0 を指定すると塗りつぶされない．

5.3.4. void dl_ellipse(int x1, int y1, int r1, int r2, double angle, double start_angle, double end_angle, long color, int thickness, int fill)

概要

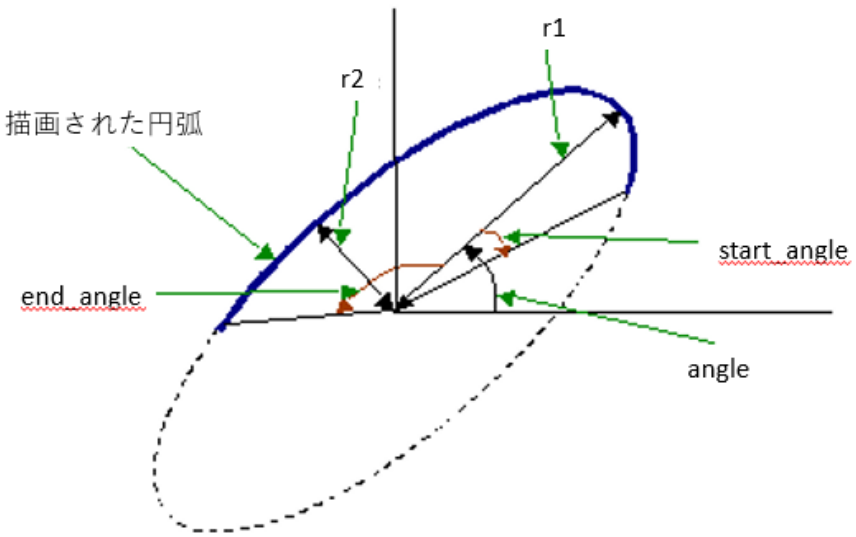
楕円，または円弧を描画する．

引数

- x1 … 中心点の x 座標
- y1 … 中心点の y 座標
- r1 … 一つ目の径の長さ
- r2 … 二つ目の径の長さ
- angle … 楕円の傾き．単位は度（0～360）
- start_angle … 開始角度（0～360）
- end_angle … 終了角度（0～360）
- color … 色に対応する整数値（dl_color_from_name 関数を参照のこと）
- thickness … 線の太さ（1 が標準．大きいほど太い線が描画される）
- fill … 1 を指定すると，塗りつぶされる．0 を指定すると塗りつぶされない．

詳細

各引数の詳細な意味は以下の通り（opencv.jp から転載・改変）．



5.3.5. void dl_text(const char *text, int x1, int y1, double scale, long color, int thickness)

概要

テキスト（文字列）を描画する.

引数

- text ... 描画する文字列の先頭アドレスをもつポインタ.
- x1 ... 描画される文字の左下の点の x 座標
- y1 ... 描画される文字の左下の点の y 座標
- scale ... 文字の大きさ．1.0 が標準的な大きさで，この値が大きいほど大きく描画される.
- color ... 色に対応する整数値（dl_color_from_name 関数を参照のこと）
- thickness ... 線の太さ（1 が標準．大きいほど太い線が描画される）

制限

文字列が描かれる範囲は描画するまで分からない．また，描画した後もプログラム的に取得することができない）.

5.3.6. void dl_clear(long color)

概要

ウィンドウ全体を塗りつぶす.

引数

- color ... 色に対応する整数値（dl_color_from_name 関数を参照のこと）

5.3.7. void dl_stop(void)

概要

描画の更新を一時的に停止する.

5.3.8. void dl_resume(void)

概要

描画の更新を再開する.

5.4. イベント処理関数

5.4.1. int dl_get_event(int *event_type, int *key, int *x, int *y)

概要

蓄積されたイベントを取り出す.

引数

- event_type ... イベントの種類を代入する変数のポインタ
- key ... キー押下イベントの際の押されたキーに対応する数値を代入する変数のポインタ
- x ... マウス操作イベントの際のマウスポインタ位置の x 座標を代入する変数のポインタ
- y ... マウス操作イベントの際のマウスポインタ位置の y 座標を代入する変数のポインタ

戻り値

イベントが取得できた場合は 1. イベントが取得できなかった場合は 0.

詳細

drawlib ウィンドウが選択されている状態でキーが押下されるか，ウィンドウ上にマウスポインタがある状態でマウスのボタンに対する操作が行われると，イベントが発生し内部的に蓄積される.

dl_get_event 関数を呼び出したときに，蓄積されたイベントがある場合，最も古いものの情報が引数に与えられたポインタが指す変数に代入され，そのイベント情報は消去される. 関数が呼び出されたときに一つもイベントが蓄積されていない場合はポインタが指す変数には何も代入されず，戻り値として 0 を出力する.

イベントの情報が取得できた場合，event_type が指す変数には以下のうちのいずれかの値が代入される.

- DL_EVENT_KEY ... いずれかのキーが押下された

- DL_EVENT_L_DOWN ... マウスの左ボタンが押された
- DL_EVENT_L_UP ... マウスの左ボタンが離された
- DL_EVENT_R_DOWN ... マウスの右ボタンが押された
- DL_EVENT_R_UP ... マウスの右ボタンが離された

イベントの種類が DL_EVENT_KEY である場合，key が指す変数には押下されたキーに対応する ASCII コード（例えば，「a」が押下された場合は 97）が代入される．特殊なキーに関しては別の値が代入されるが，そのうち有用であると思われるものについては下記の定数が定義されている．

- DL_KEY_LEFT ... 矢印キー（左）
- DL_KEY_UP ... 矢印キー（上）
- DL_KEY_RIGHT ... 矢印キー（右）
- DL_KEY_DOWN ... 矢印キー（下）
- DL_KEY_ENTER ... エンターキー
- DL_KEY_SHIFT_L ... シフトキー（左）
- DL_KEY_SHIFT_R ... シフトキー（右）

制限

引数に不正な値（ヌルポインタを含む）を与えた場合の動作は保証されない．（イベントの情報が不要でも適切な引数を与える必要がある．）

本関数を呼ばない限りイベントは蓄積され続けるため，不当にメモリ消費をする可能性がある．

5.5. 定数・マクロ

5.5.1. ウィンドウに関する定数

ウィンドウの描画できる幅，高さを表す定数．

- DL_WIDTH ... 幅．640．
- DL_HEIGHT ... 高さ．480．

5.5.2. 色に関するマクロ

色に対応する定数を得るためのマクロ．関数名が長いため，簡潔にプログラムが書けるために省略名を定義．

- DL_C(name) ... dl_color_from_name(name)と同じ動作をする
- DL_RGB(r, g, b) ... dl_color_from_rgb(r, g, b)と同じ動作をする

改訂歴

- 2021 年 4 月 15 日 rev. 2.2
    - ダウンロードする drawlib のバージョンを 1.0.2 に変更
- 2020 年 5 月 9 日 rev. 2.1
    - Word 版を作成
- 2019 年 3 月 26 日 rev. 2.0
    - ver. から rev. 表記に変更
    - Ubuntu 18.04 を標準に変更（16.04 に関する記述は削除）
- 2018 年 7 月 11 日 ver. 1.2
    - `dl_stop`, `dl_resume` 関数の説明を追加
- 2018 年 6 月 23 日 ver. 1.1
    - `dl_clear` 関数を追加
    - 定数・マクロの説明を追加
- 2018 年 6 月 20 日 ver. 1 初版