

# Simple Drawing FrameWork

Y.Nakaue

2022 年 3 月 30 日

## 1 プログラムの開始と終了

### 1.1 ライブラリ機能の初期化

```
void init()
```

引数 無し

返回值 無し

### 1.2 ライブラリ機能の終了処理

```
void quit()
```

引数 無し

返回值 無し

## 2 メインループ

### 2.1 描画内容の更新

```
bool System::update
```

引数 無し

返回值 メインループの更新可否

描画内容を最新の状態に更新する．常に true を返す．

## 3 描画のための設定

### 3.1 ウィンドウの作成

```
int32_t openWindow(uint32_t width, uint32_t height)
```

引数     width: 横幅, height: 高さ

返回值   作成したウィンドウ ID

描画を行うためのウィンドウを作成し、画面前面に表示する。引数には作成するウィンドウの横幅と高さを指定する。1つのプログラムの中で複数のウィンドウを作成することが可能で、返回值として作成したウィンドウに割り当てられたウィンドウ ID を返す。ウィンドウ ID は、作成した順に 0 からの連番で整数値が返される。

### 3.2 ウィンドウを閉じる

```
void closeWindow(int32_t win_id)
```

引数     win\_id: 閉じるウィンドウの ID

返回值   無し

## 4 主な描画関数

### 4.1 背景色を変更

```
void setBackground(Color color, int32_t win_id = 0)
```

引数     color: 背景色, win\_id: 背景色を設定するウィンドウの ID

返回值   無し

背景色を引数で指定した色に設定する。新たな背景色は、この関数によって設定した次のフレーム (System::update() が呼び出されたタイミング) で反映される。

### 4.2 文字列

```
void print(std::string str, int32_t win_id = 0)
```

引数     str: 文字列データ, win\_id: 描画先ウィンドウ ID

返回值   無し

引数に指定した文字列を描画する。第 2 引数に描画を行うウィンドウ ID を指定でき、デフォル

トではメインウィンドウへの描画を行う。1 フレーム内で複数回この関数を呼び出すと自動的に改行され、2 回目以降の呼び出し時には次の行に出力される。

### 4.3 図形の描画

```
void Shape::draw(Color color = {255, 255, 255, 255}, int32_t win = 0)
```

引数      color: 描画する色, win: 描画ウィンドウ ID

返り値    無し

図形各種は、基底クラスである Shape クラスを継承したクラスのオブジェクト（インスタンス）を作成し、そのメンバ関数である draw() を呼び出すことで描画を行うことができる。

#### 4.3.1 線分

コンストラクタ

```
Line(uint32_t x0, uint32_t y0, uint32_t x1, uint32_t y1, int32_t thickness  
= 1)
```

引数

x0: 始点の x 座標、y0: 始点の y 座標、x1: 終点の x 座標、y1: 終点の y 座標、thickness: 太さ

### 4.4 円

コンストラクタ

```
Circle(uint32_t x, uint32_t y, uint32_t r)
```

引数

x: 中心の x 座標、y: 中心の y 座標、r: 半径

---

#### ソースコード 1 使用例

```
1  #include <sdfw.h>
2
3  using namespace sdfw;
4
5  int main()
6  {
7      init();
8      openWindow(1280, 720);
9  }
```

```

10     while (System::update())
11     {
12         // (500, 500)から(600, 600)までの太さ 1の線分を描画する
13         Line(500, 500, 600, 600, 1).draw();
14
15         // (100, 100)を中心とする半径 10の円を描画する
16         Circle(100, 100, 10).draw();
17     }
18
19     quit();
20 }

```

---

## 5 イベントに関する機能

以下の機能は `System::update()` が呼び出されたタイミングで状態が更新されているため、少し古い情報が得られる場合がある。

### 5.1 マウス入力

#### 5.1.1 マウスカーソル座標の取得

`Point pos()`

引数      無し

返り値    現在のマウスカーソル座標

現在のマウスカーソルの座標値を `Point` 型で取得できる。また、`x` 座標・`y` 座標はそれぞれメンバ変数 `x`・`y` から取得できる。

---

#### ソースコード 2 使用例

---

```

1 #include <sdfw.h>
2 #include <string>
3
4 using namespace sdfw;
5
6 int main()
7 {
8     init();
9
10    openWindow(1280, 720);
11
12    std::string str;

```

```

13     while (System::update())
14     {
15         // マウ斯卡ーソルの座標をテキスト出力
16         str = "X: " + std::to_string(Mouse::pos().x) + ", Y: " + std::to_string(Mouse::pos().y);
17         print(str);
18     }
19
20     quit();
21 }

```

---

### 5.1.2 マウスボタン入力状態の取得（押されているかを取得）

`bool pressed(int8_t button)`

引数     `button`: マウスボタン（マクロ）

返り値   指定したマウスボタンが押されているか

入力状態を得たいマウスボタンの指定には、マクロによるビットマスクを使用する．ここで使用するマクロは以下の表 1 のように定義されている．また、これらをビット OR で結合して引数に指定することで、複数のボタンが同時に入力されているかを得ることもできる．

表 1   マウスボタンを表すマクロ名

ボタン	マクロ名
マウス左ボタン	LEFT
マウス中ボタン	MIDDLE
マウス右ボタン	RIGHT

ソースコード 3   使用例

```

1  // マウス左ボタンが押されているかを取得
2  sdfw::Mouse::pressed(LEFT);
3
4  // マウス左・右ボタンが両方押されているかを取得
5  sdfw::Mouse::pressed(LEFT | RIGHT);

```

---

## 6 タイマ関連機能

### 6.1 SDFW が初期化されてから経過したフレーム数を取得する

```
uint32_t Time::getTicks()
```

引数 無し

返回值 現在のフレーム数

SDFW を初期化してからのフレーム数を返す。フレーム数と時間との関係は、プログラム実行時のフレームレートに依存する。

### 6.2 SDFW が初期化されてから経過した時間（ミリ秒）を取得する

```
uint32_t Time::getMillisec()
```

引数 無し

返回值 現在の経過時間（ミリ秒）

SDFW を初期化してからの経過時間をミリ秒で返す。

### 6.3 現在フレームまでの平均 FPS 値を取得する

```
float getAverageFPS()
```

引数 無し

返回值 平均 FPS

現在のフレームまでの平均 FPS 値を取得する。

※この値の精度は低い。