理化学研究所　御中

平成 30 年度
InSitu 処理向け二次元可視化ライブラリの
プロトタイプ整備

使用説明書

平成 30 年 7 月

富士通株式会社

目次

1. はじめに

本書は、「平成 30 年 InSitu 処理向け二次元可視化ライブラリのプロトタイプ整備」作業において開発した二次元可視化ライブラリ「Pi2D」についての使用説明書です。
本書では、Pi2D ライブラリのインストールおよび使用方法について説明しています。

2. インストール方法

2.1. 動作環境

Pi2D ライブラリは以下に示す環境での動作を前提としています。

(1) OS
  Linux (CentOS 7.x / Ubuntu 16.x)
  macOS 10.13 (High Sierra)
(2) コンパイラ
  GCC 4.8 / 5.4
  LLVM 9.1
(3) Python
  Python 2.7 / 3.5 / 3.6
    numpy 1.14
    matplotlib 2.2
    six 1.11

2.2. インストール方法

2.2.1. Python モジュールのインストール

Pi2D ライブラリを動作させるには Python2.7 または Python3.x の環境と、幾つかの Python モジュールが必要です。
Python モジュールのインストールは、pip コマンドを使用して実行します。pip コマンドは、通常 Python 環境に付属してインストールされています。

(1) Python2.7 の場合
  sudo pip install numpy matplotlib six

(2) Python3.x の場合
  sudo pip3 install numpy matplotlib six

2.2.2. Pi2D ライブラリのコンパイル

Pi2D ライブラリはソースコードで提供されており、使用するにはコンパイルする必要があります。以下に、Pi2D ライブラリのコンパイル方法について説明します。

(1) Python 開発ライブラリのインストール

Pi2D ライブラリのコンパイルには、Python ライブラリの開発環境がインストールされている必要があります。Python ライブラリの開発環境がインストールされているかどうかは、以下のコマンドを実行することで確認できます。

・ Python 2.7

```
which python2.7-config
```

・ Python 3.5

```
which python3.5-config
```

・ Python 3.6

```
which python3.6-config
```

上記コマンドを実行し、python*M.N*-config のパスが表示されれば、Python ライブラリの開発環境はインストールされています。

インストールされていない場合は、以下のコマンドでインストールすることが出来ます。

・ CenOS 7.x

- Python 2.7

```
sudo yum install python-devel
```

- Python 3.6

```
sudo yum install python36-devel
```

・ Ubuntu 16.x

- Python 2.7

```
sudo apt install libpython2.7-dev
```

- Python 3.5

```
sudo apt install libpython3.5-dev
```

(2) コンパイル・インストール

まず、提供環境に含まれる Pi2D ライブラリのソースを展開します。

```
tar xvfz Pi2D_version.tar.gz
```

Pi2D ディレクトリが作成されるので、そこに移動します。

```
cd Pi2D
```

ここで、使用する Python のバージョンに応じ、以下のコマンドを実行します。

・ Python 3.6

```
make
```

・ Python 3.5

```
make PY_VER=3.5
```

・ Python 2.7

```
make PY_VER=2.7
```

コンパイルに成功すると、Pi2D ディレクトリには libPi2D.a ファイルが作成されています。

成功した場合、以下のコマンドを実行することで Pi2D ライブラリを任意の場所にインストールすることが出来ます。

```
make install PI2D_DIR="インストール先ディレクトリ"
```

ここで、「PI2D_DIR="インストール先ディレクトリ"」の指定を省略すると、$HOME/Pi2D/ 配下にインストールされます。

インストールが終了すると、〝インストール先ディレクトリ〝配下には以下に示すファイルが配置されます。

```
"インストール先ディレクトリ"
├──── LICENSE
├──── LICENSE.picojson
├──── Pi2D.py
├──── bin
|       └──── Pi2D-config
├──── include
|       ├──── LUT.h
|       ├──── Pi2D.h
|       ├──── Pi2Ddefs.h
|       ├──── pi2d_c.h
|       ├──── pi2d_f.inc
|       └──── picojson.h
└──── lib
        ├──── libPi2D_d.a
        └──── libPi2D.a
```

3. ライブラリの使用方法

3.1. 概要

Pi2D は、2D 可視化ライブラリのプロトタイプであり、ユーザープログラムにリンクし、ユーザープログラム内で 2D の配列データを入力として Pi2D ライブラリの API を呼び出すことにより、可視化画像ファイルを生成する機能を提供します。

Pi2D は C++のクラスライブラリであり、C++で記述されたユーザープログラムからの利用を前提としていますが、C または FORTRAN で記述されたユーザープログラムからの利用にも対応させるためのラッパー関数も、併せて提供しています。

3.2. C++プログラムからの使用方法

(1) ヘッダーファイルのインクルード

Pi2D ライブラリの API を使用する C++プログラムファイルでは、以下のように Pi2D クラスのヘッダーファイルをインクルードする必要があります。

```
#include "Pi2D.h"
```

(2) インスタンス作成と API 呼出し

Pi2D ライブラリの API 使用は、全て Pi2D クラスのインスタンスを介して行います。一般的には、以下のように Pi2D クラスのインスタンスを作成し、API 呼出しを行います。

```
Pi2D* pi2d = new Pi2D();          // Pi2D クラスのインスタンスを作成
pi2d->SetAttrib("imageSize=640, 480"); // 可視化画像のサイズを 640x480 に設定
pi2d->SetAttrib("arraySize=128, 128");  // 入力データサイズを 128x128 に設定
pi2d->DrawS(ColorContour, data);        // 入力データ data についてカラーコンターを描画
pi2d->Output(step);               // 画像ファイル名中のステップ番号を step にし、画像出力
delete pi2d;                      // Pi2D クラスのインスタンスを破棄
```

ここで、DrawS メソッドに引数として与える data は Real 型の配列で、サイズは 128x128 である必要があります。Real 型は、コンパイル時に_REAL_DBL シンボルが定義されていれば double 型に、未定義であれば float 型に定義されます。

インスタンス作成と API 呼出しが同一スコープ内であれば、インスタンスを一時オブジェクトとして作成し、使用することも可能です(この場合は delete は不要です)。

```
Pi2D pi2d;
pi2d.SetAttrib("imageSize=640, 480");    // pi2d はポインタではないので"."でアクセスする
...
```

利用可能な API とその仕様については、「システム設計書」の第３章「C++ API」を参照してください。

⑶ コンパイルと実行

Pi2D ライブラリを使用する C++プログラムのコンパイル・リンクおよび実行時には、以下の環境変数を設定しておく必要があります。

PY_VER	Pi2D のコンパイルに使用した Python のバージョン(未設定時: 3.6)
PI2D_DIR	Pi2D をインストールしたディレクトリ(未設定時: $HOME/Pi2D)

コンパイル・リンク時にコンパイラ・リンカに与えるオプションを Pi2D-config コマンドを使用して得ることが出来ます。Pi2D-config コマンドは、$PI2D_DIR/bin ディレクトリに存在します。

```
Usage:   Pi2D-config [--double] [--cflags] [--ldflags] [--libs] [--help]
        --double  Real 型を double として定義する(指定しなければ float として定義する)
        --cflags  コンパイラに与えるオプションフラグを表示する
        --ldflags リンカに与えるオプションフラグを表示する
        --libs    リンカに与えるリンクライブラリオプションを表示する
```

Makefile 等では、Pi2D-config コマンドの出力を参照するのに、以下のような記述を行います。

```
CXXFLAGS=`$(PI2D_DIR)/bin/Pi2D-config --cflags`
LDFLAGS=`$(PI2D_DIR)/bin/Pi2D-config --ldflags`
LIBS=`$(PI2D_DIR)/bin/Pi2D-config --libs`
```

以下のような記述にすると、Real 型を double として定義してコンパイル・リンクを行います。

```
CXXFLAGS=`$(PI2D_DIR)/bin/Pi2D-config --double --cflags`
LDFLAGS=`$(PI2D_DIR)/bin/Pi2D-config --double --ldflags`
LIBS=`$(PI2D_DIR)/bin/Pi2D-config --double --libs`
```

3.3. C プログラムからの使用方法

⑴ ヘッダーファイルのインクルード

Pi2D ライブラリの API を使用する C プログラムファイルでは、以下のように Pi2D ライブラリのラッパー関数用ヘッダーファイルをインクルードする必要があります。

```
#include "pi2d_c.h"
```

⑵ ラッパー関数の呼出し

Pi2D ライブラリの API 使用は、全てラッパー関数の呼出しを通して行います。

```
int iret;
iret = pi2d_init();
iret = pi2d_setattrib("imageSize=640,480", 17);
iret = pi2d_setattrib("arraySize=128,128", 17);
iret = pi2d_draws(0, data, "default", 7, 10, 0);
iret = pi2d_output(0, 0, 0, 0);
iret = pi2d_finalize();
```

利用可能な API とその仕様については、「システム設計書」の第５章「C/FORTRAN プログラム用 API」を参照してください。

(3) コンパイルと実行

Pi2D ライブラリを使用する C プログラムのコンパイル・リンクおよび実行時に必要な環境変数の設定および Makefile 等での Pi2D-config コマンドの出力の参照については、C++プログラムの場合と同様です。3.2 章の(3)項を参照してください。

ただし、リンカとして C コンパイラコマンドを使用する場合は、標準 C++ライブラリをリンクするようにリンカにリンクライブラリオプションを指定する必要があります。

- Linux での Makefile の記述例

```
LIBS=`$(PI2D_DIR)/bin/Pi2D-config --libs` -lstdc++
```

- macOS での Makefile の記述例

```
LIBS=`$(PI2D_DIR)/bin/Pi2D-config --libs` -lc++
```

3.4. FORTRAN プログラムからの使用方法

(1) ヘッダーファイルのインクルード

Pi2D ライブラリの API を使用する FORTRAN プログラムファイルでは、以下のように Pi2D ライブラリのラッパー関数用ヘッダーファイルをインクルードする必要があります。

```
include 'pi2d_f.inc'
```

(2) ラッパー関数の呼出し

Pi2D ライブラリの API 使用は、全てラッパー関数の呼出しを通して行います。

```
integer :: iret
character*128 :: strbuf
iret = pi2d_init()
strbuf = "imageSize=640,480"
iret = pi2d_setattrib(strbuf, 17)
strbuf = "arraySize=128,128"
iret = pi2d_setattrib(strbuf, 17)
strbuf = "default"
```

```
iret = pi2d_draws(0, data, strbuf, 7, 10, 0)
iret = pi2d_output(0, 0, 0, 0)
iret = pi2d_finalize()
```

利用可能な API とその仕様については、「システム設計書」の第５章「C/FORTRAN プログラム用 API」を参照してください。

⑶ コンパイルと実行

Pi2D ライブラリを使用する FORTRAN プログラムのコンパイル・リンクおよび実行時に必要な環境変数の設定および Makefile 等での Pi2D-config コマンドの出力の参照については、C++プログラムの場合と同様です。3.2 章の(3)項を参照してください。

ただし、リンカとして FORTRAN コンパイラコマンドを使用する場合は、標準 C++ ライブラリをリンクするようにリンカにリンクライブラリオプションを指定する必要があります。

- Linux での Makefile の記述例

```
LIBS=`$(PI2D_DIR)/bin/Pi2D-config --libs` -lstdc++
```

- macOS での Makefile の記述例

```
LIBS=`$(PI2D_DIR)/bin/Pi2D-config --libs` -lc++
```