mdspass 使用手引書

2024 年 3 月 18 日
株式会社知能情報システム
担当: 大竹遼河、松枝敦夫

[目次]

[変更履歴]

日付	版	担当	変更内容
2024/03/01	1.0.0	大竹遼河	新規作成。
2024/03/18	1.1.0	松枝敦夫	「4.1 Windows の場合」におけるビルド方法を CMakeLists.txt を使ったものに変更。

1 概要

この文書は、分子動力学シミュレーション ソフトウェア mdspass の使用手引書です。ソフトウェアのディレクトリ構成、操作方法、ビルド方法、および FLTK を用いた GUI の作成方法について記載します。

2 ディレクトリ構成

2.1 Source/ ディレクトリ

mdspass のプログラム一式を含む Source/ ディレクトリの内容を表 1 に示します。

表 1 Source/ ディレクトリの内容

ファイルまたはディレクトリ	説明
mdspass2/	ソリューション ファイルのルート ディレクトリです。
mdspass2/mdspass2.sln	ソリューション ファイルです。Visual Studio で一括してプロジェクトを管理するためのファイルです。
mdspass2/mdspass2/	プロジェクトのルート ディレクトリです。
mdspass2/mdspass2/CMakeLists.txt	CMake の設定ファイルです。
mdspass2/mdspass2/library/	ライブラリ ディレクトリです。プロジェクトで使用する外部ライブラリを格納しています。
mdspass2/mdspass2/mdspass2.vcxproj	プロジェクト ファイルです。Visual Studio でプロジェクトの構成、ビルド設定、依存関係などを管理するためのファイルです。
mdspass2/mdspass2/src/	ソース ディレクトリです。

2.2 Binary/ ディレクトリ

mdspass のプログラムを Windows でビルドした実行ファイルを含むディレクトリです。

3 操作方法

3.1 mdspass の実行

Windows の場合は、"Binary/mdspass2.exe" からプログラムを実行します。また「4.1 Windows の場合」 の手順に従ってビルドすることもできます。ご自身でビルドした場合、実行ファイルは "Source/mdspass2/x64/Release/" に配置されます。Ubuntu の場合は、「4.2 Ubuntu の場合」の手順に従いビルドしたプログラム "mdspass" から実行します。

3.2 元のプログラムから変更された操作

MD Viewer ウィンドウの表示操作のためのウィジェットをコントロール ウィンドウから削除しました。これに伴い、mdspass の画面操作はウィジェットではなく、マウス ドラッグによる操作に変更されました (表 2)。

表 2 MD Viewer ウィンドウの操作方法

操作	方法
回転	マウス右ボタンでドラッグ
平行移動	Shift キーを押しながら、マウス右ボタンでドラッグ
拡大、縮小	Control キーを押しながら、マウス右ボタンでドラッグ

4 ビルド方法

mdspass のビルド方法を OS ごとに説明します。以下の手順で各 OS にてビルドしてください。また、ビルドに必要なライブラリを表 3 に示します。

表 3 mdspass のビルドに必要なライブラリ

ライブラリ名	バージョン	配布 URL
freeglut	3.4.0	https://freeglut.sourceforge.net/
zlib	1.3	http://zlib.net/
libpng	1.6.40	http://www.libpng.org/pub/png/libpng.html
CLAPACK	3.1.1	https://www.netlib.org/clapack/
FLTK	1.3.8	https://www.fltk.org/

4.1 Windows の場合

ビルド環境を表 4 に示します。

表 4 Windows でのビルド環境

OS	Windows 10 Pro
統合開発環境	Visual Studio 2022

Windows でのビルド手順を以下に示します。

1. git のインストール
   git を公式ページ（https://gitforwindows.org/）からダウンロードし、インストールしてください。
2. vcpkg のインストール
   Windows 用の C++ パッケージ マネージャーである vcpkg をインストールします。Windows PowerShell を起動し、vcpkg をインストールしたいディレクトリに移動してから、以下のコマンドを実行してください。（ディレクトリのパスに日本語文字列と半角スペースを含むとインストールに失敗することがありますので、ご注意ください。）

```
> git clone https://github.com/microsoft/vcpkg.git
```

   インストールが完了したら、以下のコマンドを実行してください。

```
> cd vcpkg
> .¥bootstrap-vcpkg.bat
```

   vcpkg.exe という名前の実行ファイルがローカル リポジトリ内に生成されていれば、vcpkg のインストールは完了です。
3. ライブラリのインストール
   vcpkg をインストールしたディレクトリで以下のコマンドを実行し、必要なライブラリ一式（FLTK、LAPACK、BLAS）をインストールしてください。

```
> .¥vcpkg install fltk blas lapack
```

4. Visual Studio からライブラリにアクセス
   integrate コマンドで、vcpkg でインストールしたライブラリを Visual Studio から利用できるようにします。

```
> .¥vcpkg integrate install
```

5. Visual Studio 2022 の起動
   Visual Studio 2022 を起動し、起動時の画面で［コードなしで続行］をクリックします（図 1 の赤丸）。

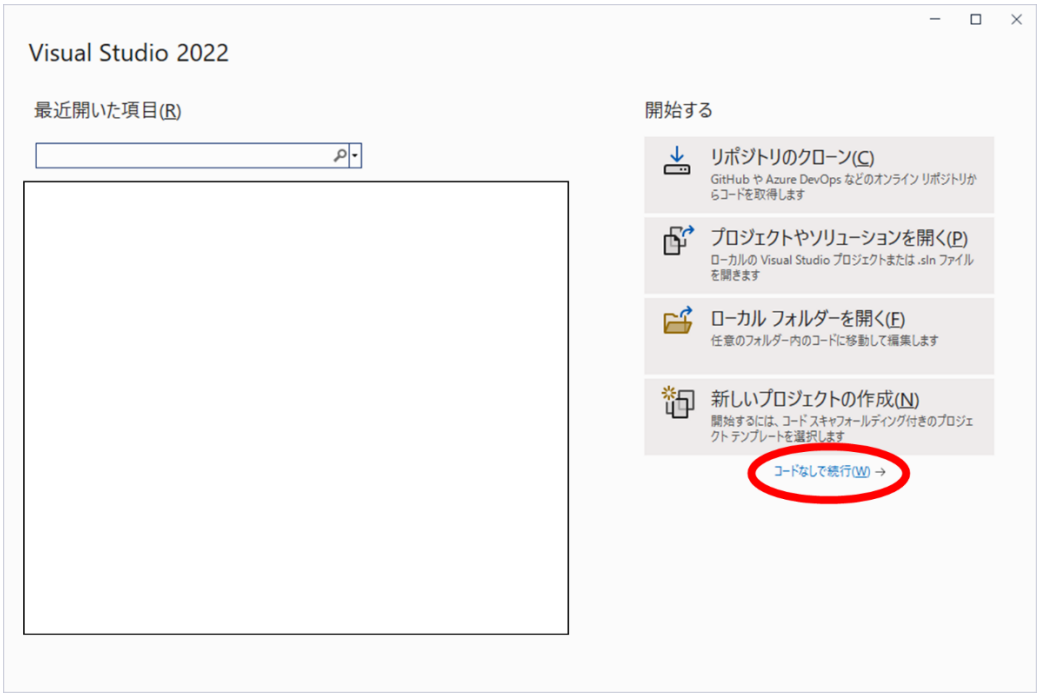

図 1 Visual Studio 2022 起動時の画面

6. CMakeLists.txt を開く

   [ファイル] > [開く] > [CMake] メニューから、mdspass の CMakeLists.txt ファイルを開きます。

7. CMake の設定画面を開く

   ツール バー上の "x64-Debug" と表示されているドロップダウン リスト ボックス（図 2 赤丸）から、"構成を管理します" を選択し、CMake の設定画面を開きます。

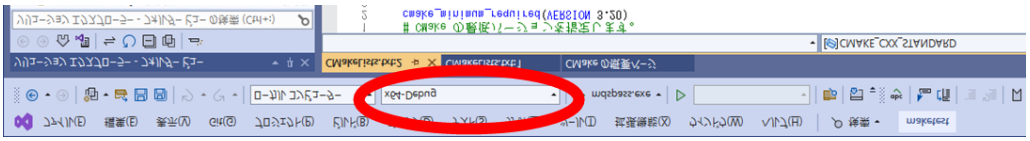

図 2 ツール バー

8. Release ビルドの設定を追加

   CMake の設定画面の [構成] にある設定のクローンを行うボタン（図 3 赤丸 1 の中の一番右のボタン）をクリックして、"x64-Debug" の構成のクローンを作成します。作成された設定を選んだ後、[構成名]（図 3 赤丸 2）を "x64-Release" に修正し、[構成の種類]（図 3 赤丸 3）から "Release" を選択します。その後、キーボード ショートカット Ctrl + S を押下して、設定ファイルを保存、反映します。

図 3 CMake の設定画面

9. Release に切り替えてビルド

ツール バーの上のドロップダウン リスト ボックス（図 2 の赤丸）から先ほど追加した構成 "x64-Release" を選択します。自動的に Visual Studio 2022 が CMakeLists.txt ファイルからビルド設定を構築しますので、その完了を待ちます。その後、[ビルド] > [すべてビルド] メニューを選んで、mdspass プログラムをビルドします。CMakeLists.txt ファイルがあるフォルダーに .″¥out¥build¥x64-Release″ フォルダーが新規に作成され、その中にビルドされたプログラムが配置されています。

4.2 Ubuntu の場合

ビルド環境を表 5 に示します。

表 5 Ubuntu でのビルド環境

OS	Ubuntu 22.04

Ubuntu でのビルド手順を以下に示します。以下の作業は、クリーン インストールした Ubuntu で、ネットワーク接続が可能な環境、sudo が可能なユーザーであることを想定しています。

1. ライブラリのインストール

ターミナルを起動し、必要なライブラリをインストールするため、以下のコマンドを実行してください。

```
$ sudo apt install cmake libfltk1.3-dev libglu1-mesa-dev liblapack-dev libblas-dev
 libpng-dev
```

2. Makefile の作成

Source/mdspass2/mdspass2/ ディレクトリの CmakeLists.txt ファイルと /src ディ

レクトリを任意のディレクトリにコピーします。コピーしたディレクトリに移動してから、以下のコマンドで Makefile ファイルを作成します。

```
$ cmake -S . -B build
```

3. ビルドの実行
   build ディレクトリに移動し、make コマンドでビルドを実行します。実行ファイル mdspass が build ディレクトリ内に生成されます。

```
$ make
```

5 FLTK を用いた GUI の作成方法

FLTK を用いて、GUI を作成、編集する方法を以下に示します。各ウィジェットについてのより詳細な情報は、公式ページ（https://www.fltk.org/doc-1.3/index.html）をご参照ください。

5.1 基本的な使用方法

FLTK を用いた基本的な GUI 作成方法を以下に示します。

1. セットアップ
   FLTK ライブラリをインストールし、プロジェクトに組み込みます。
2. ウィンドウの作成
   Fl_Window クラスを使用してメイン ウィンドウを作成します。
3. ウィジェットの追加
   ボタン、テキスト ボックスなどのウィジェットをウィンドウに追加します。
4. イベント処理
   ウィジェットのアクションやユーザー入力に応じてイベントを処理するコールバック関数を定義します。
5. ウィンドウの表示と実行
   show メソッドでウィンドウを表示し、Fl::run メソッドでイベント ループを開始します。

以下に、FLTK でボタンを表示するサンプル コードと、コードを実行した際の GUI 画面を示します（図 4）。

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>
```

```
// コールバック関数です。
void button_callback(Fl_Widget* widget, void*) {
    // ボタンがクリックされたときのアクションです。
    widget->label("Clicked!");
}

int main() {
    // ウィンドウを作成します。
    // コンストラクターの引数は、ウィンドウの幅、高さ、タイトルです。
    Fl_Window* window = new Fl_Window(300, 200, "FLTK Example");
    // ウィンドウの設定を開始します。
    window->begin();

    // ボタンを追加します。
    // 引数は、x 座標、y 座標、幅、高さ、ラベルです。
    Fl_Button* button = new Fl_Button(50, 100, 100, 30, "Click Me");
    // コールバックを登録します。
    button->callback(button_callback);

    // ウィンドウの設定を終了します。
    window->end();
    // ウィンドウを表示します。
    window->show();

    // FLTK イベント ループを実行します。
    return Fl::run();
}
```

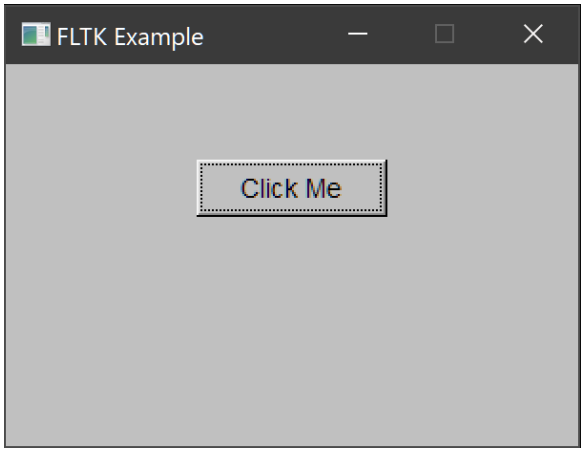

図 4 サンプル コード実行時の GUI 画面

5.2 FLTK のウィジェット

FLTK では様々なウィジェットが用意されています。mdspass で使用した FLTK のウィジェットの一覧を表 6 に示します。

表 6 mdspass に使用した FLTK のウィジェット

FLTK のウィジェット	機能
Fl_Box	ボックス
Fl_Button	ボタン
Fl_Check_Button	チェック ボックス
Fl_Choice	ドロップダウン リスト
Fl_File_Browser	ファイル ブラウザ
Fl_File_Chooser	ファイル選択ダイアログ
Fl_Gl_Window	OpenGL 描画用ウィンドウ
Fl_Group	複数のウィジェットをグループ化するウィジェット
Fl_Input	テキスト ボックス
Fl_Pack	ウィジェットを一列に並べるウィジェット
Fl_Radio_Round_Button	ラジオ ボタン
Fl_Spinner	スピナー

5.3 mdspass に GUI を追加する方法

以下の手順で mdspass に GUI を追加することができます。

1. ウィンドウの作成
   Fl_Window クラスか BaseWindow クラス（スクロールやサイズ変更などを設定済みの Fl_Window の派生クラス）を継承したクラスを作成してください。既存のウィンドウに GUI を追加する場合は、この手順は不要です。
2. ウィンドウの呼び出し
   MainWindow クラス、あるいは呼び出し元のウィンドウのクラスで、新たに作成したウィンドウのインスタンスを生成します。create_window_open_button 関数を用いて、ウィンドウを呼び出すボタンを作成することができます。
3. ウィジェットの追加
   ウィジェットを追加したいウィンドウのファイルを開き、ウィジェットを追加します。表 6 に示したウィジェットに関しては、custom_gui_functions.h に定義されている関数を使用して追加することができます。

以下に、mdspass に新しいウィンドウ（NewWindow）を追加し、呼び出し元のウィンドウにウィンドウ表示ボタンを作成するサンプル コードを示します。

NewWindow クラスのヘッダー ファイル

```
#pragma once

#include "BaseWindow.h"

/**
* サンプルのウィンドウ クラスです。
**/
class NewWindow : public BaseWindow
{
public:
    /* コンストラクターです。*/
    NewWindow();

    /**
     * イベント ハンドラー
     * @param event イベント識別子
     **/
    int handle(int event) override;
};
```

NewWindow クラスのソース ファイル

```
#pragma once

#include "custom_gui_functions.h"

// ウィジェットと紐づける変数です。
int mock1 = 0;
int mock2 = 0;

NewWindow::NewWindow() : BaseWindow(300, 300, "New Window")
{
    // ウィジェットを配置する列を用意します。
    Fl_Pack* column1 = new Fl_Pack(0, 0, 150, 150);

    // 列の設定を開始します。
    column1->begin();
    // チェック ボックスを作成します。
    create_check_button(150, 30, "label1", &mock1);
    // int 型の入力を作成します。
    create_int_input(150, 80, "label2", &mock2);
    // ウィンドウを閉じるボタンを作成します。
    create_window_close_button(80, "Close");
    // 列の設定を終了します。
    column1->end();

    end_window();
}
```

```
int NewWindow::handle(int event)
{
    switch (event)
    {
    // ウィンドウが表示された時の処理です。
    case FL_SHOW:
        // 表示時の処理（値の初期化など）を記述します。
        return 1;

    // ウィンドウが閉じられた時の処理です。
    case FL_HIDE:
        // 処理（値の初期化など）を記述します。
        return 1;

    default:
        return Fl_Window::handle(event);
    }
}
```

呼び出し元のウィンドウでの、新しいウィンドウ表示ボタンの作成

```
#include <memory>
#include "NewWindow.h"
#include "custom_gui_functions.h"

// NewWindow クラスのインスタンスを生成します。
new_window = std::make_unique<NewWindow>();

// ウィンドウを表示するボタンを追加します。
// 引数には、ボタンの幅と、ラベル、表示するウィンドウのインスタンスを指定します。
create_window_open_button(100, "New Window", new_window)
```