# Springhead Library インストールガイド (Windows 版)

第 1.0 版 – 2021/03/17

Springhead Library のインストールとビルドの方法、及びサンプルプログラムのビルド方法について説明します。

改 訂 履 歴

第 1.1 版　ドキュメント名称変更
第 1.0 版　初版

# 目次

# 1　インストール

まず、Springhead Library のダウンロード及びビルドの方法について説明します。

## 1.1　ダウンロード

Springhead Library は GitHub で管理されており、次の URL からダウンロード (クローン) することができます。

"https://github.com/sprphys/Springhead"

---

Springhead Library のクローン及びビルドには git が必要です。予めインストールしておいてください。
参考 URL :

| | |
|---|---|
| git | https://gitforwindows.org/ |
| GUI ツール | https://tortoisegit.org/ |

---

以下、ダウンロードするディレクトリを "C:\Springhead" として説明を進めます。

Explore 上で右クリックし “Git Clone...” を選びます (図 1)。
Git clone の画面で、URL に https://github.com/sprphys/Springhead を、Directory にダウンロードする場所 (この例では "C:\Springhead" ) を指定します。さらに recursive にチェックを入れ、OK を押します (図 2)。以上で、Springhead Library 及びそのサブモジュールがダウンロードされます。

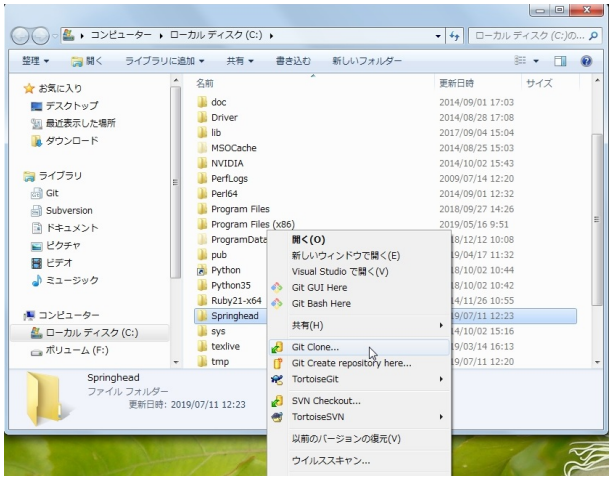


図 1　Git Clone... の選択

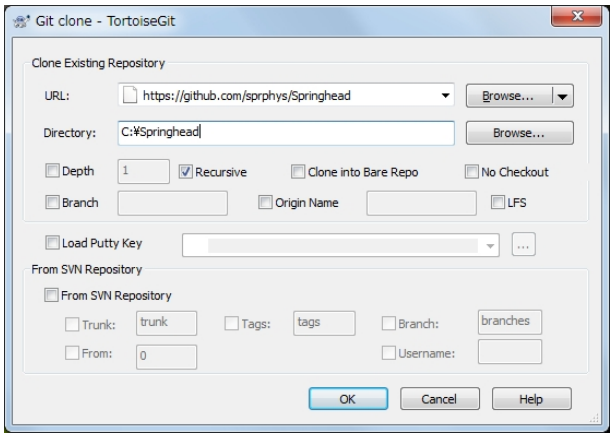


図 2　URL 等の指定

コマンドラインからダウンロードする場合は、次のようにします。

```
> chdir C:/Springhead
> git clone --recurse-submodules  (次の行に続く)
                    https://github.com/sprphys/Springhead
```

次に `http://git.haselab.net/haselab/springheadclosed.git` から closed ソースを同様にクローンして `"C:\Springhead\closed"` に置きます。
以上でダウンロードは終了です。

**(注意)** Springhead Library のビルドには `python` 及び `nkf` が必要です。buildtool サブモジュールをダウンロードしていない場合には、別途インストールしてパスに加えておいてください。

## 1.2　Springhead Library のビルド

Springhead Library のビルド方法について説明します。ただし、サンプルプログラムをビルドする場合に限りこの節での作業は不要です。

ディレクトリ `"C:\Springhead\core\src"` へ移動します。

複数のソリューションファイルがあります。使用する Visual Studio のバージョンに合ったものを使用してください。

---

`"Springhead`*nn.n*`.sln"` の *nn.n* の部分がバージョンを表します。

`14.0` — 2015 用
`15.0` — 2017 用
`16.0` — 2019 用

---

ソリューションファイルを Visual Studio で起動したら、Springhead を “スタートアッププロジェクト” に指定してビルドします。ライブラリファイルは、

"C:\Springhead\generated\lib\win64"
または
"C:\Springhead\generated\lib\win32"
のいずれかに生成されます。

ライブラリファイル名は次のようになります。

| 構成名 | ライブラリファイル名 | ビルド設定 |
|---|---|---|
| Release | Springhead##.lib | multithread, DLL |
| Debug | Springhead##D.lib | multithread, Debug, DLL |
| Trace | Springhead##T.lib | multithread, Debug, DLL |

- ##は、Visual Studio バージョン及びプラットフォームを表す Win32 又は x64 との組み合わせとなります ( "15.0x64" など).
- Trace 構成とは，フレームポインタ情報付き Release 構成のことです.

## 1.3 EmbPython のビルド

### 1.3.1 Springhead アプリケーションに Python インタプリタを組み込む場合

ディレクトリ "C:\Springhead\core\src\EmbPython" に移動します。

適切なバージョンのソリューションファイル "EmbPython *nn.n* .sln" を Visual Studio 起動し、ターゲット EmbPython をビルドします。
ライブラリファイルは、
"C:\Springhead\core\src\EmbPython"
に次の名前で生成されます。

| 構成名 | — | ライブラリファイル名 |
|---|---|---|
| Relase | — | EmbPython##.lib |
| Debug | — | EmbPython##D.lib |
| Trace | — | EmbPython##T.lib |

- ##については前節と同様

### 1.3.2 Python インタプリタに対する外部拡張モジュール (Python DLL, pyd) を作成する場合

ディレクトリ "C:\Springhead\core\embed" に移動します。
適切なバージョンのソリューションファイル "SprPythonDLL *nn.n* .sln" を VisualStudio で起動し、ターゲット SprPythonDLL をビルドします。DLL ファイルは、
"C:\Springhead\generated\bin\x64"
または
"C:\Springhead\generated\bin\x32"
に次の名前で生成されます。

| 構成名 | — | DLL ファイル名 |
|---|---|---|
| Relase | — | Spr.pyd |
| Debug | — | SprD.pyd |
| Trace | — | SprT.pyd |

# 2　サンプルプログラムのビルド

サンプルプログラムのビルド方法について説明します。

サンプルプログラムは、"C:\Springhead\core\src\Samples" 以下に置かれています。ここでは “BoxStack” を例にして説明します。

"C:\Springhead\core\src\Samples\Physics\BoxStack" に移動します。
適切なバージョンのソリューションファイル "BoxStack*nn.n*.sln" を Visual Studio で開き、プロジェクト BodStack をスタートアッププロジェクトに設定してビルドすれば実行形式バイナリが生成されます。

実行時に DLL が見つからないというエラーが発生した場合には、

| | | |
|---|---|---|
| 32 ビット環境のときは | — | "C:\Springhead\dependency\bin\win32" |
| 64 ビット環境のときは | — | "C:\Springhead\dependency\bin\win64"<br>"C:\Springhead\dependency\bin\win32"<br>の両方 |

にパスを通してください。Visual Studio から実行するときは、プログラムのプロパティを開き、[構成プロパティ]—[デバッグ]—[環境] に “path=上記のパス” とします。

また、実行時に Assertion エラーが発生し

“You have to define USE_CLOSED_SRC in SprDefs.h to use Spidar”

などと表示された場合には、"C:\Springhead\core\include\SprUseClosedSrcOrNot.h" にある #undef を #define と変更し、サンプルプログラムを再ビルドしてください。

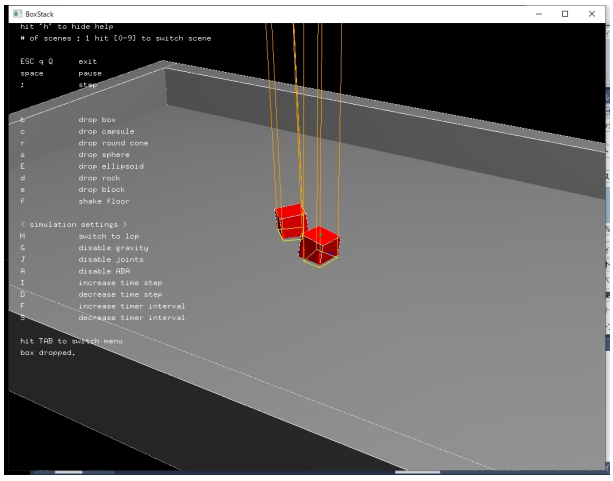


図 3　実行結果