

Blender をコンパイルする

1. 目的

これまで開発環境としてきた ノート PC から デスクトップ PC に環境を変更するにあたって `import bpy` を実行しようとするエラーが発生し、`pip install bpy` でも目的のパッケージを見つけられなかったため検索をかけたところ、以下のことがわかった。

参考: [【超入門編】Blender Python \(BPY\) の使い方](#)

bpy: Blender の様々な機能を Python で簡単に呼び出せるようにするため、Blender 開発者が公開している橋渡しの役割を持つモジュール（関数・構造体の集まり）のことです。この橋渡しの役割を持つモジュールは一般に API と呼ばれています。

Blender 内の python を用いて操作をする場合、必ず冒頭に"import bpy"と記述する必要があります。これを最初に記述しておくことで bpy と名のついた関数を実行できるようになります。

そして、Blender をビルドすることで、インストール済みの python に組み込むことができるらしい [参考: [Blender のビルド手順 その9 \(Python モジュールへの Blender2.8 の組み込み\)](#)]。

よって、まず Blender をビルドしていく。

2. ビルド環境を整える

まず、ビルドするための環境を整える。その際、[\[Blender\] ソースコードから Blender 本体をビルドする](#) を参考にした。





2.1. Visual Studio Community 2019 のインストール

Blender のコアの部分は C 言語と C++ 混在で書かれているため、Blender をビルドするためには、Windows 上でビルドするための環境が必要です。

そのために、Microsoft が提供している IDE である [Visual Studio 2019](#) をインストールする。

インストール中に、追加でダウンロードするパッケージを選択できるが、今回は **C++によるデスクトップ開発** のみを選択した。

デスクトップとモバイル (5)

 .NET デスクトップ開発 C#、Visual Basic、F# を使用して、WPF、Windows フォーム、コンソール アプリケーションをビルド...	<input type="checkbox"/>	 C++ によるデスクトップ開発 MSVC、Clang、CMake、MSBuild など、選択したツールを使用して、Windows 用の最新の C++ アプリをビルドします。	<input checked="" type="checkbox"/>
 ユニバーサル Windows プラットフォーム開発 C#、VB、または C++ (オプション) を使ってユニバーサル Windows プラットフォームのアプリケーションを作成します。	<input type="checkbox"/>	 .NET によるモバイル開発 Xamarin を使用して iOS、Android、または Windows 向けのクロスプラットフォーム アプリケーションをビルドします。	<input type="checkbox"/>

2.2. SVN のインストール

Blender のビルドには、コンパイル済みのライブラリが必要になります。Git はバイナリのバージョン管理には不向きであるため、コンパイル済みのライブラリは SVN で管理されています。

ということで、Blender のコンパイル済みライブラリをダウンロードするために、SVN をインストールする。

最新の SlickSVN は以下の URL から インストールできる。

<https://sliksvn.com/download/>

svn をインストール後は、exe ファイルにパスを通す必要がある。

例: x 64 をインストールした場合:

```
$ C:\Program Files\SlikSvn\bin\svn.exe
```

をパスに追加し、コマンドプロンプトにて以下のような結果になることを確認する。

```
$ svn
# Type 'svn help' for usage.
```

2.3. ソースコード・バイナリの取得

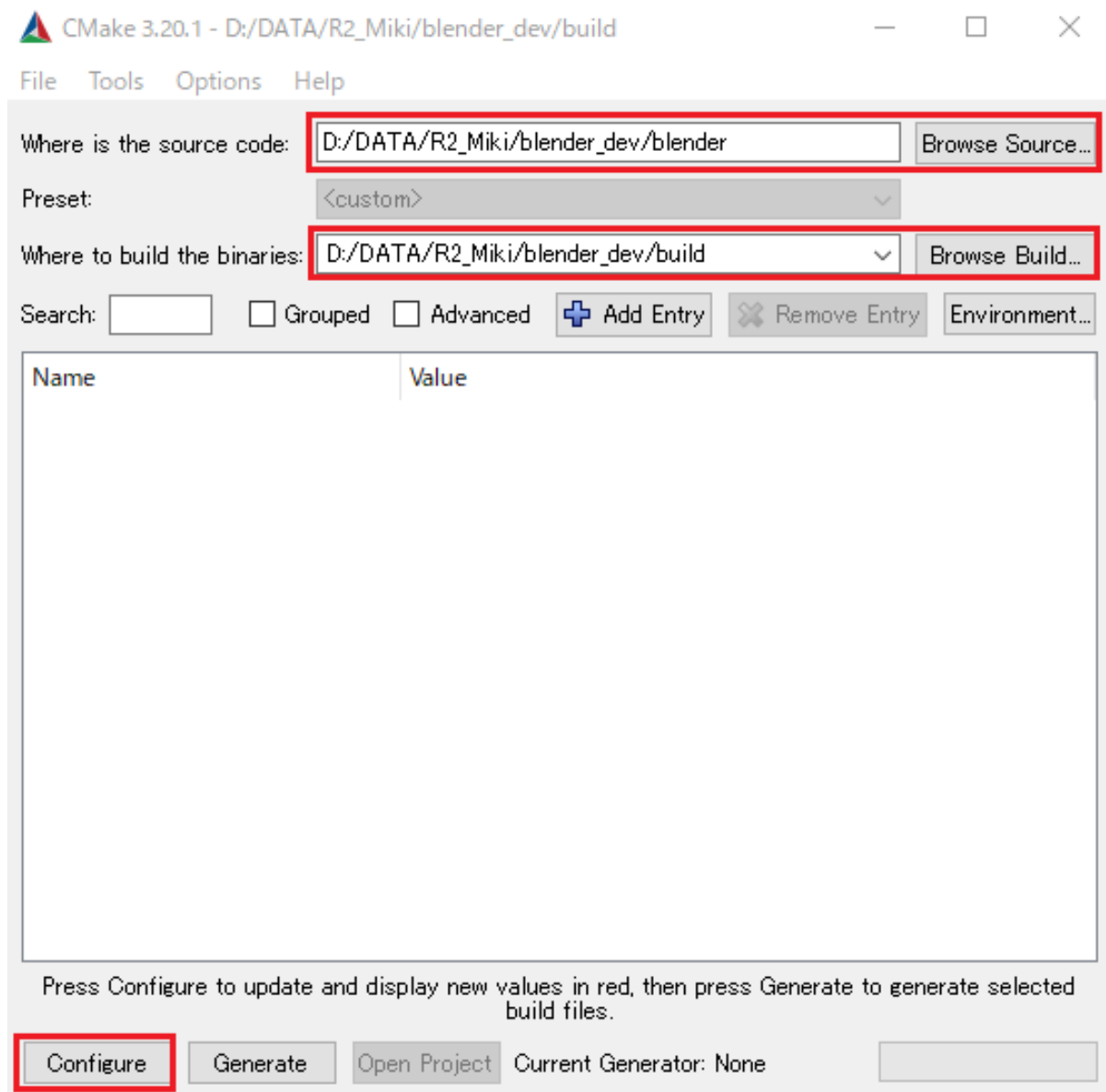
```
$ mkdir blender_dev
$ cd blender_dev
$ git clone https://github.com/blender/blender.git
$ cd blender
$ git submodule update --init --recursive
$ git submodule foreach git checkout master
$ git submodule foreach git pull --rebase origin master
$ cd ..
$ svn checkout https://svn.blender.org/svnroot/bf-blender/trunk/lib/win64_vc12
lib/win64_vc12
```

2.4. プロジェクトファイルの作成

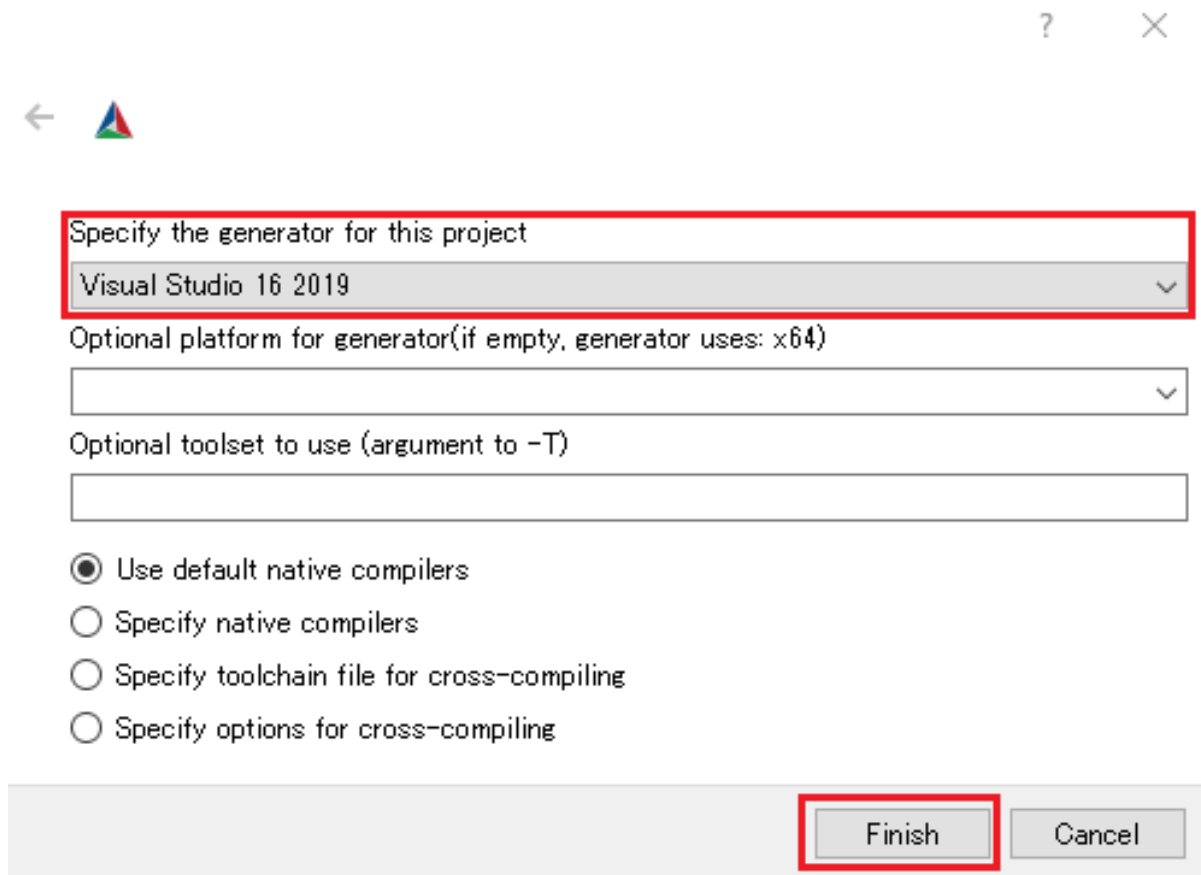
ビルド後の Blender のバイナリを置くためのディレクトリを作成する。

```
$ cd blender_dev
$ mkdir build
```

CMake を起動する。ソースが置かれたディレクトリ と ビルドによって作られるデータを置くディレクトリを指定し、**configure** をクリックする。



続いて表示されるウィンドウでは、インストールした **Visual Studio** を指定して **Finish** をクリック。



2.5. エラー発生

cmake を実行すると、次のようなエラーが発生した。

```
Windows requires pre-compiled libs at:
'D:/DATA/R2_Miki/blender_dev/blender/../../lib/win64_vc15'. Please run `make
update` in the blender source folder to obtain them.
Call Stack (most recent call first):
CMakeLists.txt:936 (include)
```

このエラーについては、以下のサイトに解決方法が記載されていた。

2.5.1. make をインストールする

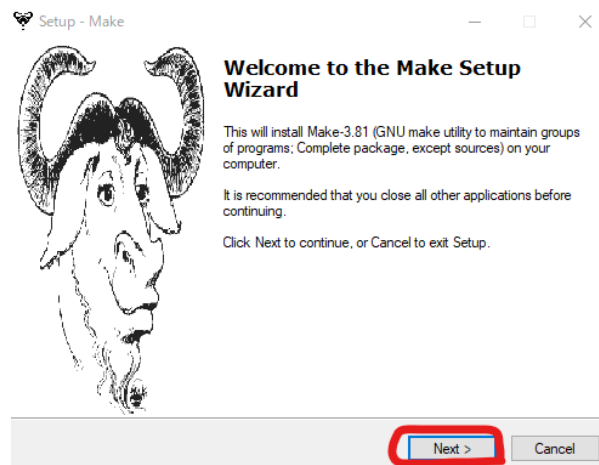
まず、次のサイトにしたがって、**make** をインストールする。

[Windows 環境で make コマンドを利用する](#)

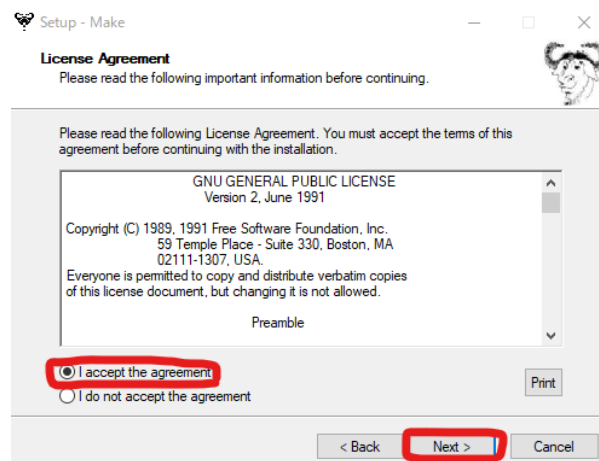
1. [make の公式サイト](#) にアクセスし、**Complete package, except sources Setup** をクリックする。

Download				
<p>If you download the Setup program of the package, any requirements for running applications, such as dynamic link libraries (DLL's) from the dependencies as listed below under Requirements, are already included. If you download the package as Zip files, then you must download and install the dependencies zip file yourself. Developer files (header files and libraries) from other packages are however not included; so if you wish to develop your own applications, you must separately install the required packages.</p>				
Description	Download	Size	Last change	Md5sum
• Complete package, except sources	Setup	3384653	25 November 2006	8ae51379d1f3eef8360df4e674f17d6d
• Sources	Setup	1252948	25 November 2006	b896c02e3d581040ba1ad65024bbf2cd
• Binaries	Zip	495645	25 November 2006	3521948bc27a31d1ade0dcb23be16d49
• Dependencies	Zip	708206	25 November 2006	d370415aa924fa023411c4099ef84563
• Documentation	Zip	2470575	25 November 2006	43a07e449d4bab3eb3f31821640ecab7
• Sources	Zip	2094753	25 November 2006	8bed4cf17c5206f8094f9c96779be663

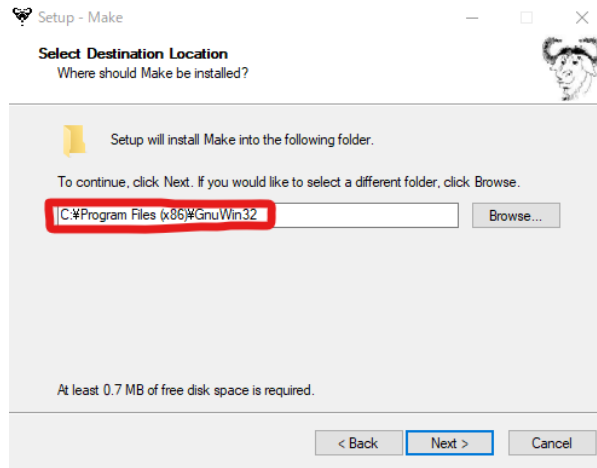
2. ダウンロードした **make.exe** を立ち上げ、**Next**



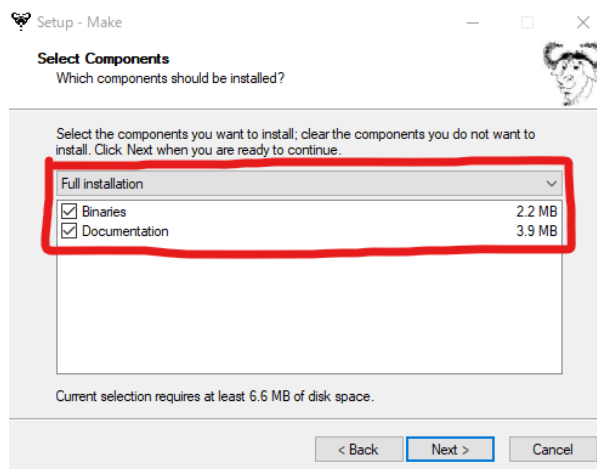
3. I accept the agreement -> **Next**



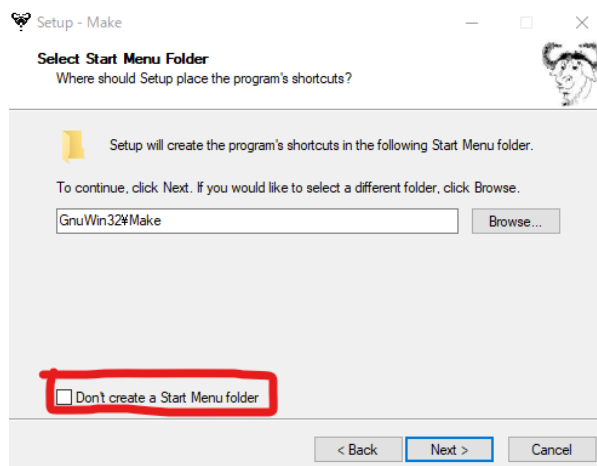
4. インストール先を指定



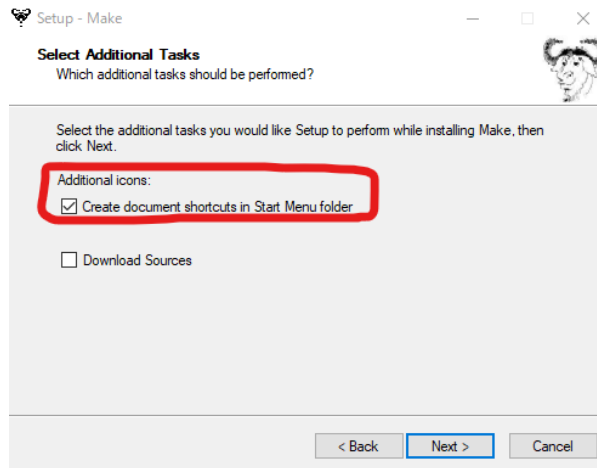
5. インストールするコンポーネントを選択



6. スタートメニューの設定を行う。スタートメニューを作りたくない場合は、**Don't create a Start Menu folder** にチェックを入れる。今回は、スタートメニューを作る方を選択した。



7. その他のオプション設定を行う。**Create document shortcuts** 特に指定が無ければ、そのまま **Next** をクリックする。



8. インストールしたディレクトリを環境変数に指定する

```
$ C:\Program Files (x86)\GnuWin32\bin
```

2.5.2. make update

先ほどのエラーは、`make update` ができていないために発生しているようだった。そこで、以下のコマンドで blender を make update する

```
$ cd blender_dev
$ cd blender
$ make update
```

2.5.3. エラー対処完了

`make update` を実行することで、CMake の **Configure** を完了できた。成功すると、以下のように、**Configuring done** と表示される。

```
Disabling render tests because tests folder does not exist at D:/DATA/R2_Miki/blender_dev/blender/../../lib/tests
Configuring done
```

これで、一応ビルドが完了したことになる。

2.6. プロジェクトファイル作成続き

Configuring done と表示された後に、** Generate** をクリックする。しばらく時間が経った後、以下のようになり **Generating done** という文字が表示されるとプロジェクトファイルの作成に成功している。

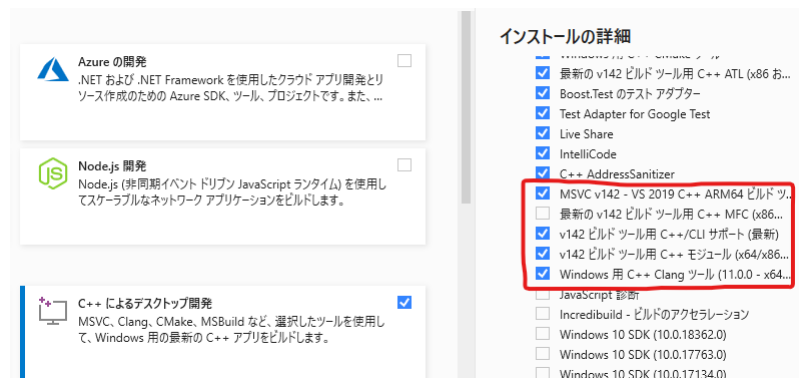
```
Disabling render tests because tests folder does not exist at D:/DATA/R2_Miki/blender_dev/blender/../../lib/tests
Configuring done
Generating done
```

2.7. Blender をビルドする

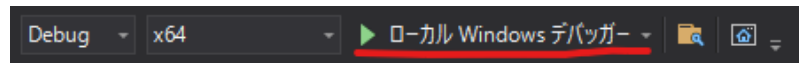
CMake でビルドの設定を行った際に指定した、ビルド時のデータが置かれるディレクトリに移動する。プロジェクトファイルの作成に成功している場合、**Blender.sln** ファイルがある。

tests	2021/07/20 9:57	ファイル フォルダー
x64	2021/07/20 9:58	ファイル フォルダー
ALL_BUILD.vcxproj	2021/07/20 9:57	VC++ Project
ALL_BUILD.vcxproj.filters	2021/07/20 9:57	VC++ Project Filte
blender.crt.manifest	2021/07/20 9:57	MANIFEST ファイル
blender.exe.manifest	2021/07/20 9:57	MANIFEST ファイル
Blender.sln	2021/07/20 9:57	Visual Studio Solu
cmake_install.cmake	2021/07/20 9:57	CMAKE ファイル
CMakeCache.txt	2021/07/20 9:57	テキストドキュメント

このファイルを開く前に、Visual Studio Installer を開き、以下のパッケージを追加しておく



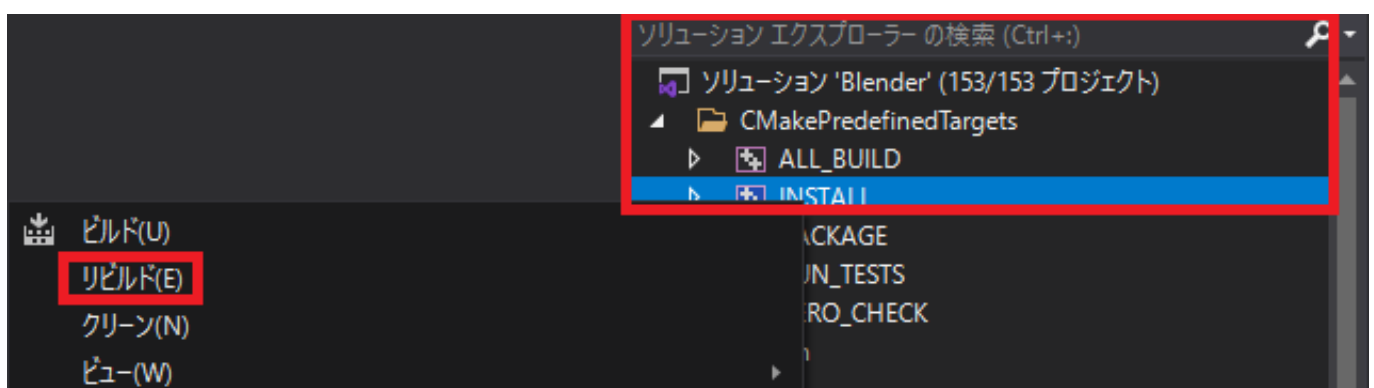
パッケージ追加後、**Blender.sln** ファイルを開き、**ローカル Windows デバッガー** を実行する。



実行後、デバッガーのエラーが表示されるが、**ビルド: 143 正常終了、0 失敗、0 更新不要、0 スキップ** が表示されれば成功。なお、ビルド数は Blender のバージョンによって異なるので、**0 失敗** となっていることが重要。

```
143> ライブラリ D:/DATA/R2_Miki/blender_dev/build/bin/Debug/blender.lib とオブジェクト D:/DATA/R2_Miki/blender_dev/build/bin/Debug/blender.exp を作成中
143>blender.vcxproj -> D:/DATA/R2_Miki/blender_dev/build/bin/Debug/blender.exe
===== ビルド: 143 正常終了、0 失敗、0 更新不要、0 スキップ =====
```

次に、ソリューションエクスプローラーの検索欄に **INSTALL** と入力し、表示された **INSTALL** されたものを右クリックして、**リビルド** を選択する。



リビルドに成功すると、ビルドと同様に **0 失敗** と表示され、Build -> bin -> Debug に blender.exe が作成される。


```
151>-- Installing: D:/DATA/R2_Miki/blender_dev/build/bin/Debug/3.0/datafiles/usd/usdVol/resources/usdVol
151>-- Installing: D:/DATA/R2_Miki/blender_dev/build/bin/Debug/3.0/datafiles/usd/usdVol/resources/usdVol/schema.usda
===== すべてリビルド: 151 正常終了、0 失敗、0 スキップ =====
```

3. モジュールとして Blender をビルドする。

次の記事を参考に、先ほどとは少し異なる方法で Blender をビルドしていく。今回は、[Blender のモジュールビルドとインストール\(Windows\)](#) を参考にした。

3.1. CMake オプションの変更

「2.4. プロジェクトファイルの作成」と同様にフォルダをセットし、CMake のオプションを以下のように変更する。この時のフォルダ選択は、これまでの手順と同様のものを使用する。

- WITH_PYTHON_INSTALL=OFF
- WITH_PYTHON_MODULE=ON

ビルドが失敗する原因になるライブラリを取り除く

- WITH_AUDASPACE=OFF
- WITH_OPENCOLLADA=OFF (リンクエラー xml.lib dllmain libcmtd.lib)

```
WITH PYTHON INSTALL
WITH_PYTHON_INSTALL_NUMPY
WITH PYTHON MODULE
WITH_PYTHON_NUMPY
```



ここまで、設定したら

1. Configure
2. generate**
3. .sln ファイルを Visual Studio で開く
4. ローカル Windows デバッカー を実行する。
5. ビルドが成功した場合、ソリューションエクスプローラーの検索欄に **INSTALL** と入力し、表示された **INSTALL** されたものを右クリックして、**リビルド** を選択する。

3.2. ビルドできた **bpy.pyd** と必要なファイルを Python3 に対してインストール

4. python 仮想環境構築 (venv を用いる場合)

PVN3D を用いようと git からダウンロードしたディレクトリで `pip install pip3 install -r requirements.txt` を実行したところエラーが発生したので、仮想環境を構築し改めてインストールを試みる。

4.1. 環境構築

Python3.3 以上では、パッケージ環境（仮想環境）の切り替えとして venv が標準で使えます。

そこで、まず現在インストールされている python のバージョンを確認する。

参考: [Python:venv + pyenv での環境構築](#)

```
$ python -V  
  
Python 3.8.5
```

次に、以下のコマンドでカレントディレクトリに仮想環境を構築する。

```
$ python -m venv .venv
```

そして、以下のコマンドで仮想環境を立ち上げる。

参考: [Windows で venv で仮想環境を作成して立ち上げるまでやる](#)

```
$ .\.venv\Scripts\activate
```

4.1.1. pip をアップデートする

pip を用いてパッケージをインストールする前に、以下のコマンドで pip をアップデートしておく。

```
$ py -m pip install --upgrade pip
```

4.1.2. パッケージインストール

以下のコマンドで必要なパッケージを一括でインストールすることができる。

```
$ pip3 install -r requirements.txt
```

4.1.3. インストールエラー

上記のコマンドでパッケージのインストールを実行したところ、以下のようなエラーが発生した。

```
ERROR: Could not find a version that satisfies the requirement yaml  
(from versions: none) ERROR: No matching distribution found for yaml
```

このエラーは、`yaml` パッケージが `pyyaml` と名称変更されていたため、パッケージを見つけられなかったことが原因であった。

参考: [Python 用の yaml パッケージをインストールするにはどうすればよいですか？](#)

4.2. python-pcl インストール

必要なパッケージをすべてインストールしたら、次に python-pcl をダウンロードしていく。

ポイントクラウドライブラリ（または PCL）は、2D / 3D 用の大規模なオープンプロジェクトです。

参考: [PointCloudLibrary](#)

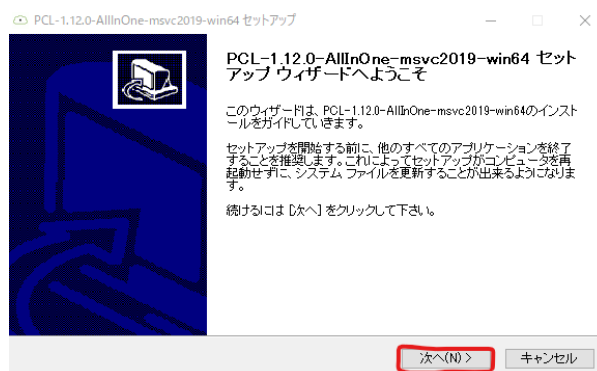
1. まず [davidcaron / pclpy](#) からサポートしている PLC のバージョンを確認する。



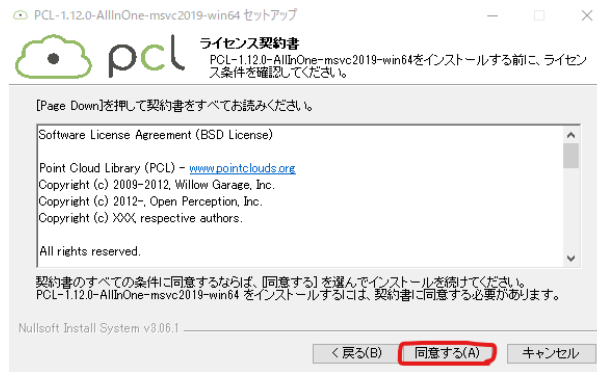
2. 次に、[PointCloudLibrary / pcl](#) から先ほどと同じバージョンの PLC で、かつ Visual Studio のバージョン、Windows 32 ビット版か Windows 64 ビット版かでダウンロードするものを選択する。



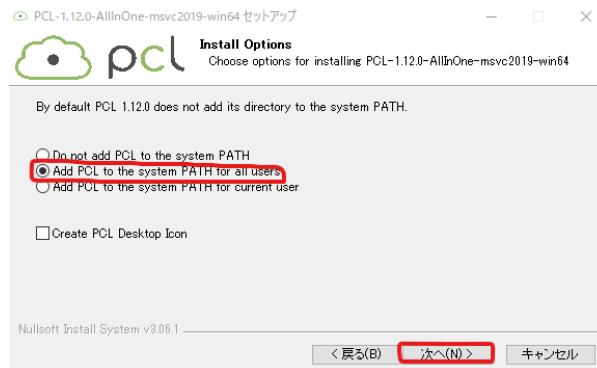
1. ダウンロードした .exe を実行する。



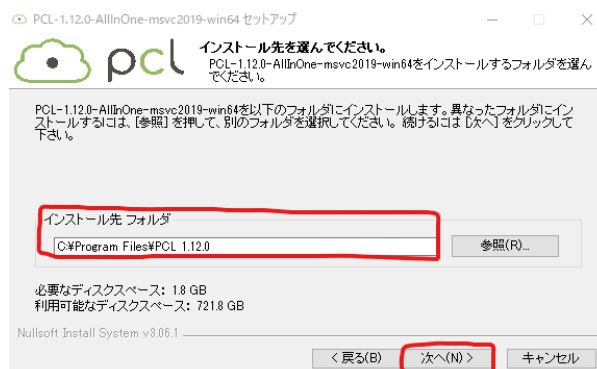
3. 同意するを選択する。



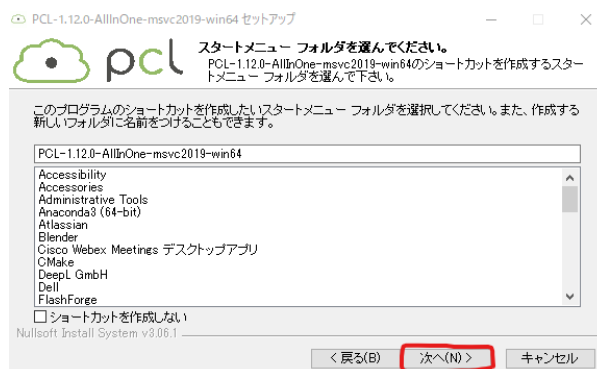
4. Add PCL to the system PATH for all users を選択し、次へ



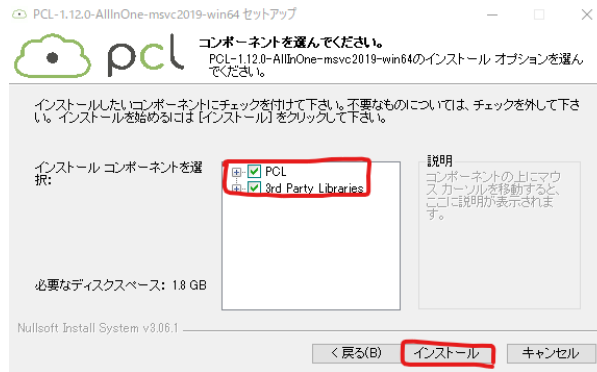
5. インストール先を選択し、次へ



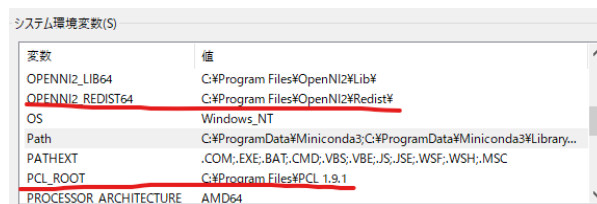
6. ショートカットを作成したいスタートメニュー フォルダを選択する。特に何もせず次へを選択。



7. インストールしたいコンポーネントを選択。今回は特に変更せず、インストールを選択。



8. インストールが完了したら、システム環境変数に **PCL_ROOT** と **OPENNI2_REDIST64** が追加されていることを確認する。



もし、ない場合は手動で以下の環境変数を設定する。なお、「1.9.1」のところは実際にインストールしたバージョンに読み替えること。

- PKG_CONFIG_PATH
 - 変数名: PKG_CONFIG_PATH
 - 変数値: C:\Program Files\PCL 1.9.1\lib\pkgconfig
- PCL_ROOT
 - 変数名: PCL_ROOT
 - 変数値: C:\Program Files\PCL 1.9.1
- OPENNI2_INCLUDE64
 - 変数名: OPENNI2_INCLUDE64
 - 変数値: C:\Program Files\OpenNI2\Include\
- OPENNI2_LIB64
 - 変数名: OPENNI2_LIB64
 - 変数値: C:\Program Files\OpenNI2\Lib\
- OPENNI2_REDIST64
 - 変数名: OPENNI2_REDIST64
 - 変数値: C:\Program Files\OpenNI2\Redist\

9. さらにシステム環境変数の **Path** に以下の 5 つを追加する。

- %PCL_ROOT%\bin
- %OPENNI2_REDIST64%

- %PCL_ROOT%\3rdParty\FLANN\bin
- %PCL_ROOT%\3rdParty\Qhull\bin
- %PCL_ROOT%\3rdParty\VTK\bin

%PCL_ROOT%bin
%OPENNI2_REDIST64%
%PCL_ROOT%\3rdParty\FLANN\bin
%PCL_ROOT%\3rdParty\Qhull\bin
%PCL_ROOT%\3rdParty\VTK\bin

4.3. davidcaron/pclpy のインストール

python 用 pcl ライブラリを作成するために、davidcaron/pclpy をインストールする。

1. まず空のインストール用ディレクトリを作る。

例えば、

```
$ mkdir c:\pytools
$ cd c:\pytools
$ rmdir /s /q pclpy
```

2. [davidcaron / pclpy](#) から、davidcaron/pclpy をダウンロードする。

```
$ cd c:\pytools
$ git clone --recursive https://github.com/davidcaron/pclpy
```