多地点遠隔 VR 可視化アプリケーション MULTI-LOCATION-PBVR

ユーザーマニュアル

Ver. 1.0

国立研究開発法人日本原子力研究開発機構システム計算科学センター

2024/02/29

目次

更新履歴

版	更新日	更新内容
1.0	2024/02/29	(初版)

1. はじめに

本書は日本原子力研究開発機構システム計算科学センター(CCSE)で開発した多地点 VR 可視化アプリケーション Multiple Location PBVR (ML-PBVR)の、インストールおよび使用方法を説明したマニュアルである。ML-PBVR は遠隔地にあるスーパーコンピュータ上の大規模シミュレーションを、計算と同時に手元の PC 上で可視化できる。そして、その可視化をスーパーコンピュータに接続した複数の PC で共有できる。ML-PBVR はヘッドマウントディスプレイ(HMD)を利用した VR 可視化が可能で、可視化パラメータを複数ユーザが変更することでユーザ間での対話的な解析が可能である。

昨今のスーパーコンピュータでは演算速度が I/O 速度を大幅に上回るため、計算結果の出力が困難になった。その結果、遠隔地にあるストレージ上の計算結果を手元の PC に転送して可視化する従来の可視化手法は適用が困難になった。In-Situ 可視化は、シミュレーションに結合された可視化プログラムが計算と同時に可視化処理を実行することで大規模なデータ I/O を回避し、確実に可視化画像を生成することができる。ポリゴンベースによる従来の手法では対話的な In-Situ 可視化が難しかった。CCSE では京都大学小山田研究室で開発された粒子ベースの可視化手法である PBVR（Particle Based Volume Rnderring）と対話的 In-Situ 制御技術により、対話的な可視化が可能な In-Situ 可視化フレームワークである IS-PBVR を開発した。そして IS-PBVR を多地点向けに拡張することで ML-PBVR を構築した。ML-PBVR は粒子サンプラ、デーモン、PBVR クライアントの３つのコンポーネントで構成されている。ML-PBVR は複数のユーザがそれぞれデーモンを起動することで、ストレージ上のファイルを介して同じ可視化データを共有できる。

(1) 粒子サンプラ

粒子サンプラはシミュレーションコードに結合され、計算と同じ環境で粒子を生成する可視化ライブラリである。シミュレーション＋In-Situ 可視化のコードを構築するには、粒子サンプラが提供する可視化用関数に計算結果の配列を渡しシミュレーションコードに挿入する。粒子サンプラはストレージ上の可視化パラメータファイルを参照して、各タイムステップの計算結果を圧縮された可視化用粒子に変換し、ストレージ上に出力する。粒子ファイルは各プロセスから分散して出力される。

(2) デーモン

デーモンはログインノードあるいは対話ジョブ上で動作し、ストレージ上の粒子ファイルと PBVR クライアントから送信されてくる可視化パラメータを仲介する機能をもち、対話的 In-Situ 制御の要となるアプリケーションである。各ユーザがログインノード上で起動する。デーモンはストレージ上のファイルを監視して分散出力された粒子ファイルを集約し、ネットワークを介してユーザ PC に転送する。またデーモンは PBVR クライアントから送信されてくる可視化パラメータを受信し、粒子サンプラが参照する可視化パラメータファイルとして出力する。この動作はシミュレーションと非同期に実行されるため、シミュレーションを阻害せずに対話的制御が可能であ

る。

(3) PBVR クライアント

PBVR クライアントはユーザ PC 上で動作し、可視化結果を表示するビューワと可視化パラメータを編集する GUI を提供する。各ユーザが起動する。PBVR クライアントはポートフォワードを使用してデーモンと通信する。PBVR クライアントはデーモンからの粒子データを受信してビューワ上にボリュームレンダリング画像を表示し、ユーザが GUI 上で編集した可視化パラメータをデーモンに送信する。

1.1. 動作環境

ML-PBVR は以下の環境で動作を確認している。

	必要スペック
CPU	Intel i3-6100 / AMD Ryzen 3 1200 / AMC FX4350
GPU	NVIDIA GTX 1050Ti / AMD Radeon RX 470
メモリ	8GB RAM
HMD	Oculus Rift (CV1) / Oculus Rift S / Quest / Quest2
OS	Windows 10 (Home/Pro) / Windows 11 Pro
コンパイラ	Visual Studio 2017 / Visual Studio 2019
開発環境	Qt Creator 5.9 / Qt Creator 6.2

2. 依存ライブラリ

ML-PBVR のビルド・実行には下記に示す複数の依存ライブラリが必要である。それらのうち KVS2.9mod4meta および CGFormatExt4KVS は github(https://github.com/CCSEPBVR/KVS)
からダウンロード可能である。それ以外のライブラリは、上記 github に含まれる ReadMe.txt を参照すること。

2.1. KVS2.9mod4meta

KVS2.9mod4meta は ML-PBVR 向けに修正された KVS であり、HMD Meta 向けに基本的な可視化機能を提供する。

2.2. CGFormatExt4KVS

CGFormatExt4KVS は 3DS 形式および FBX 形式のポリゴンデータを VR-KVS 形式のポリゴンデータに変換するライブラリである。

2.3. assimp-5.0.0

Assimp は 3DS 形式のポリゴンデータの読み込み機能を提供する C++ライブラリである。

これは CGFormatExt4KVS のビルド時に静的リンクされるため、実行時には不要である。

2.4. fbx20195_fbxsdk_vs****_win.exe

これは Autodesk 社が提供する FBX 形式のポリゴンデータの読み込み機能を提供するライブラリ FBX SDK のインストーラである。Visual Studio のバージョンは 2017/2019 で動作を確認している。FBX SDK は CGFormatExt4KVS 及び ML-PBVR のビルドと実行に必要である。

2.5. freeglut-MSVC-3.0.0-2.mp

これは Windows 用の GLUT である OpenGLUT のバイナリパッケージであり、VR-KVS 及び ML-PBVR のビルドと実行に必要である。

2.6. glew-2.1.0-win32

これは Windows 用の GLEW バイナリパッケージであり、VR-KVS 及び ML-PBVR のビルドと実行に必要である。

2.7. ovr_sdk_win_1.30.0_public

Oculus SDK は ML-PBVR のビルドに必要である。

2.8. cmake-3.16.3-win64-x64

Assimp のビルドに cmake 3.16 以上が必要である。

2.9. imgui-1.79.zip

Dear imgui は HMD 上で 2D GUI を生成するオープンソース C++ライブラリであり、ML-PBVR のビルドに必要である。

3. セットアップ

本章では、ML-PBVR をコンパイルするための事前設定と手順、及び ML-PBVR の実行に必要なフィルタ及びサーバプログラムの事前設定につて説明する。これらの操作は最初に 1 回実行するだけでよい。

以下では、作業の起点となるディレクトリを<BASE_DIR>と表記する。

3.1. Visual C++の設定

始めに Visual C++ 2017/2019(Community Edition 以上が必要)を Microsoft の Web サイト(https://visualstudio.microsoft.com/ja/vs/older-downloads/)からダウンロードし、インストールする。次に Visual Studio Installer を起動し、「詳細▼」→「変更」(図 3.4.1-1)を選択する。

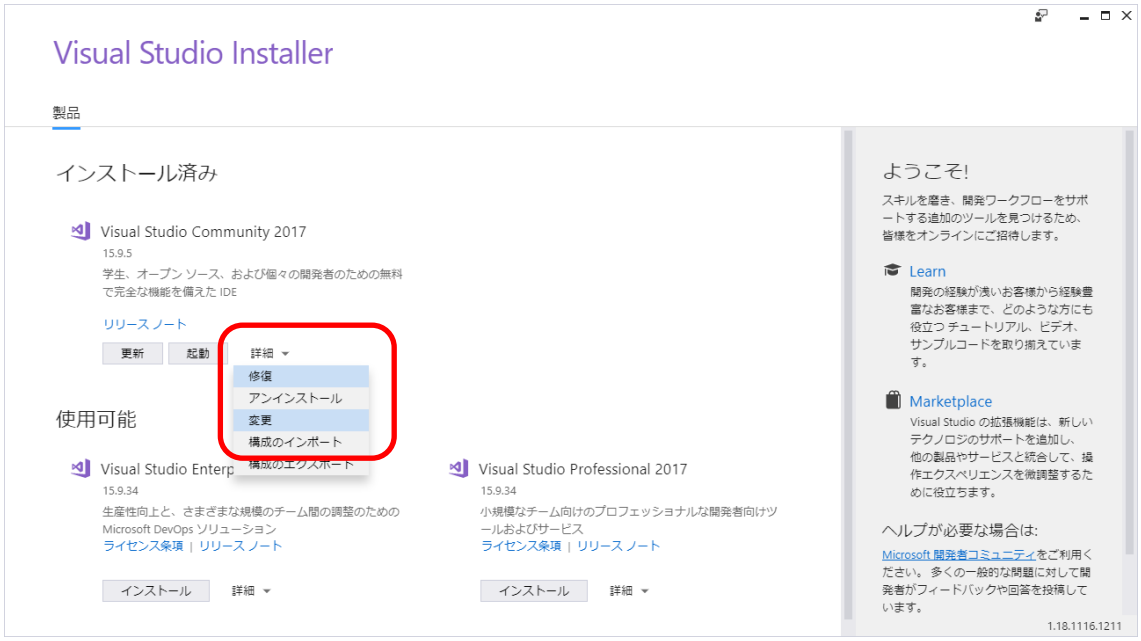


図 3.4.1-1 Visual Studio Installer の画面（1）「詳細▼」→「変更」

コンポーネントの選択画面で、以下のコンポーネントをインストールする(マイナーバージョンは違ってもよい)。

(A) 「ワークロード」タブで選択

(1) C++ によるデスクトップ開発（図 3.4.1-2)

(B) 「個別のコンポーネント」タブで選択

(1) Windows 10 SDK (10.0.17763.0) (図 3.4.1-3)

(2) デスクトップ C++ 用 Windows 10 SDK (10.0.16299.0)[x86 および x64] (図 3.4.1-3)

(3) CMake の Visual C++ ツール(図 3.4.1-4)

(4) VC++ 2017/2019 (図 3.4.1-4)

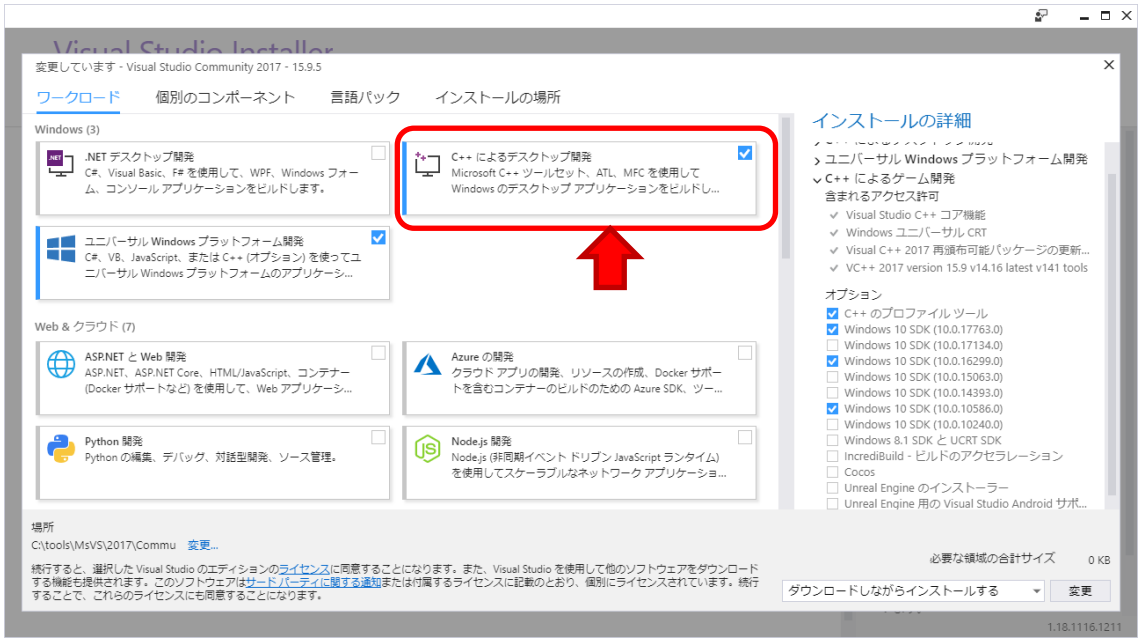


図 3.4.1-2 Visual Studio Insteller の画面（2）コンポーネント選択 - C++によるデスクトップ開発

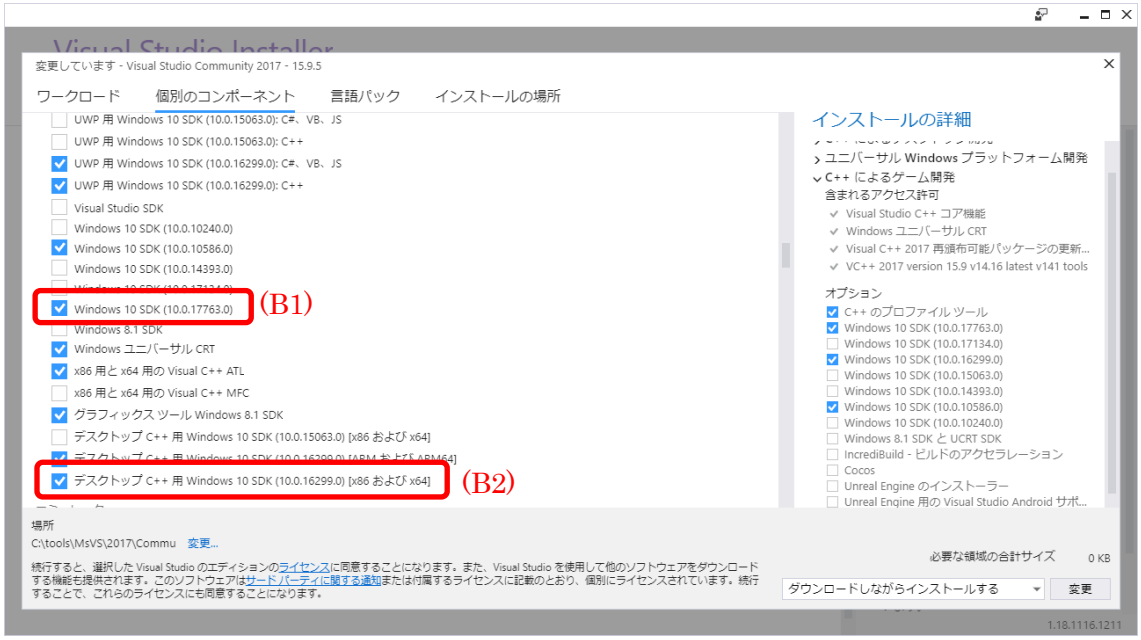


図 3.4.1-3 Visual Studio Installer の画面（3）コンポーネント選択 - Windows SDK

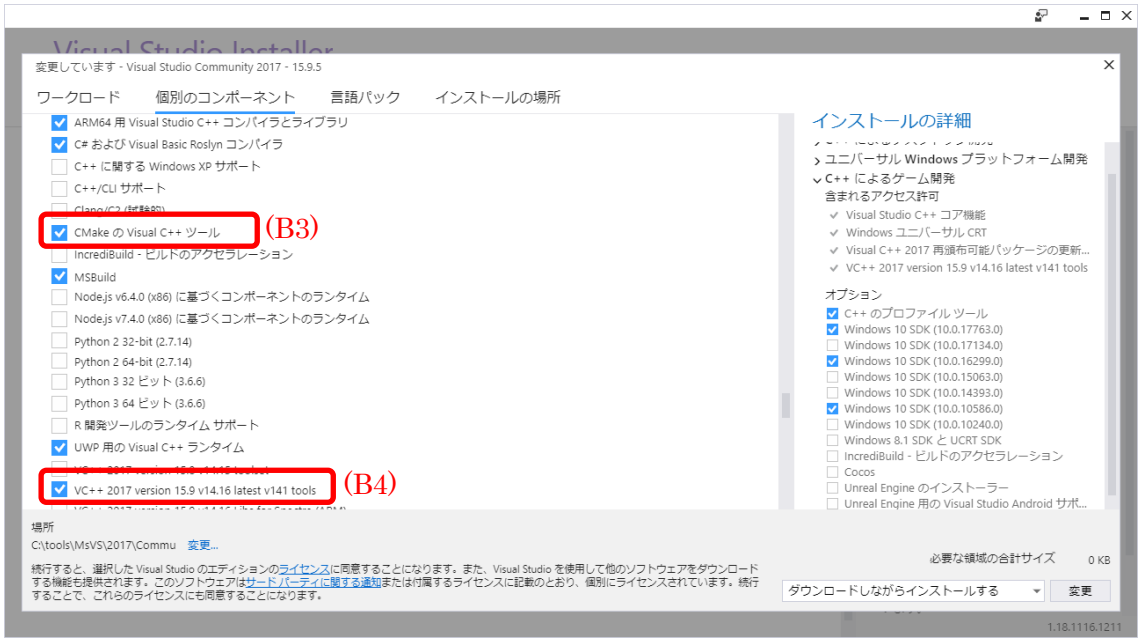


図 3.4.1-4 Visual Studio Installer の画面（3）コンポーネントの選択 - cmake と VC++ tools

コンポーネントの選択後、右下の「変更」ボタンを押下しインストールする。

3.2. Qt Creator の設定

Qt 公式 Web サイトから Qt オープンソース版のインストーラー(Qt Online Installer https://qt.io/download-open-source)をダウンロードして実行する。

Select categories において、「latest release」と「Preview」のチェックを外し、「LTS」のみチェックされた状態にする(図 3.4.1-1)。

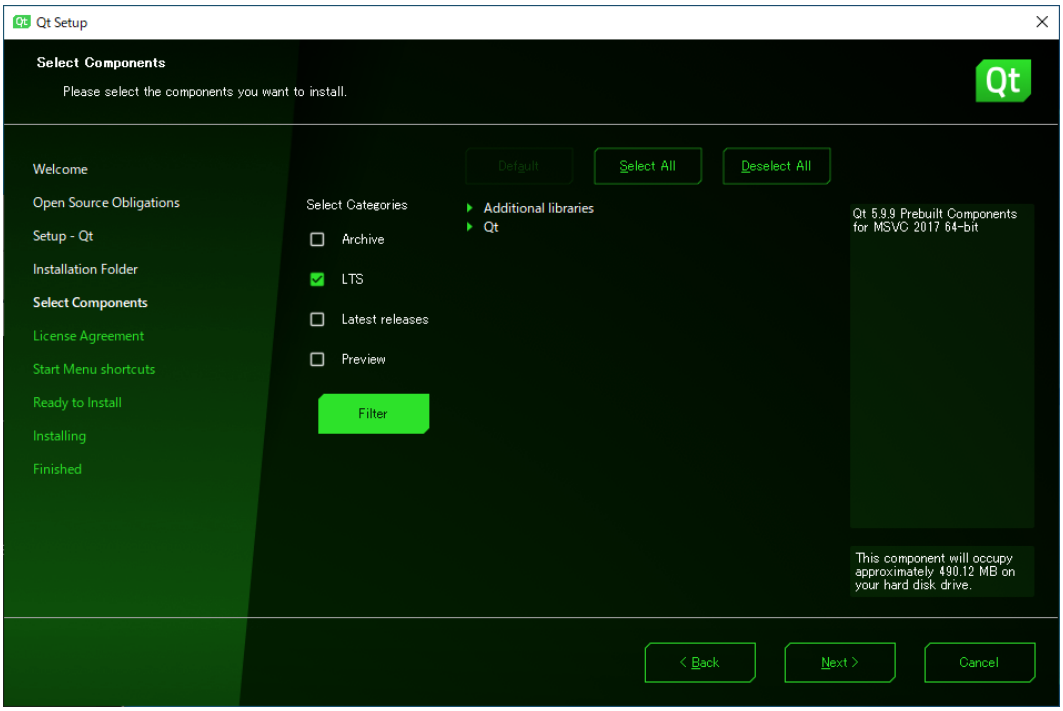


図 3.4.1-1 Qt インストーラ画面 :コンポーネントの選択(その 1)

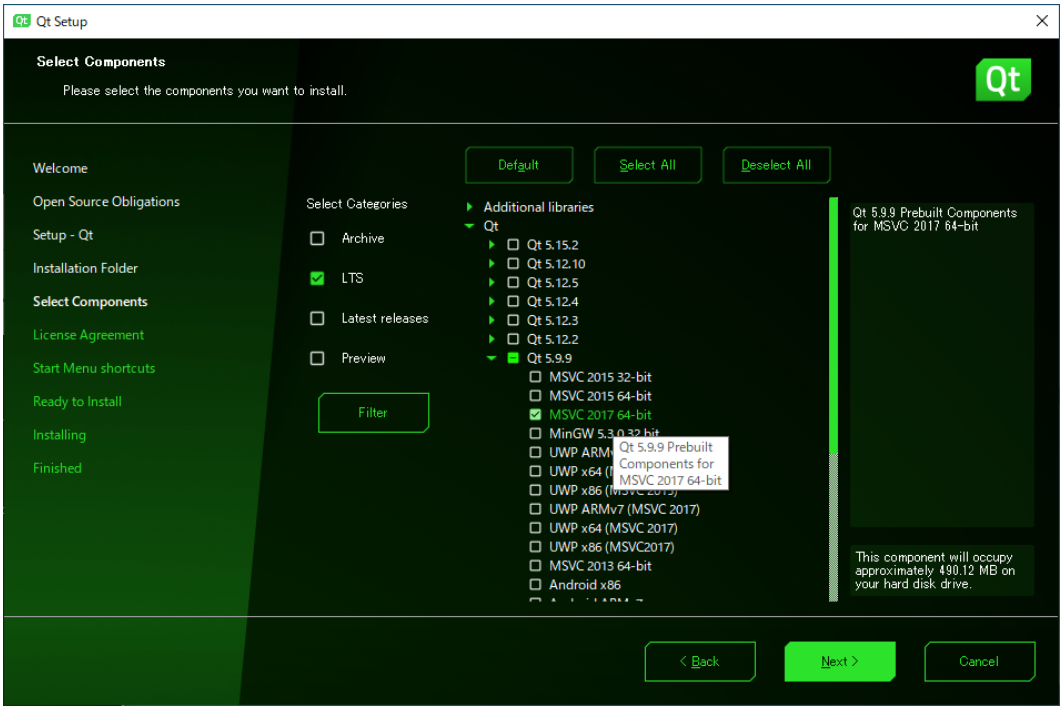


図 3.4.1-2 Qt インストーラ画面 :コンポーネントの選択(その 2)

続いて、「▸ Qt」をクリックし、下表のコンポーネントを選択する(図 3.4.1-2)。

- Qt > Qt 5.9.9 > MSVC 2017/2019 64-bit

- Qt > Qt 5.9.9 > Sources
- Qt > Developer and Designer Tools > Qt Creator 4.14.0
- Qt > Developer and Designer Tools > Qt Creator 4.13.2 CDB Debuger Support
- Qt > Developer and Designer Tools > Debugger Tools for Windows
- Qt > Developer and Designer Tools > CMake 3.18.3 64-bit
- Qt > Developer and Designer Tools > ninja 1.10.0

3.3. CMake

cmake-3.16.3-win64-x64.zip を「cmake-3.16.3-win64-x64」フォルダに解凍し、それを「<BASE_DIR>¥tools¥」フォルダに移動する。

3.4. KVS2.9mod4meta と依存ライブラリ

3.4.1. FreeGLUT

freeglult-MSVC-3.0.0-2-mp.zip を「freeglut」フォルダに解凍し、それを「<BASE_DIR>¥lib¥」フォルダに移動する。「freeglut¥lib¥x64¥」フォルダ中の「freeglut.lib」を「<BASE_DIR>¥lib¥」フォルダにコピーする。

ユーザー環境変数「KVS_GLUT_DIR」に値「<BASE_DIR>¥lib¥freeglut」を設定する。

3.4.2. GLEW

glew-2.1.0-win32.zip を「glew-2.1.0」フォルダに解凍し、それを「<BASE_DIR>¥lib¥」フォルダに移動する。「glew-2.1.0¥lib¥Release¥x64¥」フォルダ中の「glew32.lib」および「glew32s.lib」を「<BASE_DIR>¥lib¥」フォルダにコピーする。

ユーザー環境変数「KVS_GLEW_DIR」に値「<BASE_DIR>¥lib¥glew-2.1.0」を設定する。

3.4.3. KVS2.9mod4meta

ユーザー環境変数「KVS_DIR」に値「<BASE_DIR>¥lib¥kvs」を設定する。KVS はこのフォルダにインストールされる。

KVS2.9mod4meta.zip を「KVS2.9mod4meta」フォルダに解凍し、それを「<BASE_DIR>」フォルダに移動する。VS2017/2019 用 x64 Native Tools コマンドプロンプト(以下「Tools コマンドプロンプト」)を起動する。Tools コマンドプロンプト上で「<BASE_DIR>¥KVS2.9mod4meta」フォルダに移動し、以下のコマンドを実行する。

```
nmake
nmake install
```

ユーザー環境変数「Path」に値「%KVS_DIR%¥bin」を追加する。

3.5. CGFormatExt4KVS と依存ライブラリ

3.5.1. Assimp

assimp-5.0.0.zip を「assimp-5.0.0」フォルダに解凍し、それを「<BASE_DIR>¥lib¥」フォルダに移動する。

Tools コマンドプロンプトで「<BASE_DIR>¥lib¥assimp-5.0.0」フォルダに移動し、以下のコマンドを実行する。Visual Studio のバージョンは適宜読み替えること。

```
> cd assimp-5.0.0

> SET CMAKE_BIN=<BASE_DIR>¥tools¥cmake-3.16.3-win64-x64¥bin¥cmake.e
  xe

> SET SOURCE_DIR=.

> SET GENERATOR=Visual Studio 15 2017

> SET BINARIES_DIR="./BINARIES/x64"

> SET CMAKE_GENERATOR=Visual Studio 15 2017

> SET CMAKE_GENERATOR_INSTANCE=C:¥Program Files (x86)¥Microsoft Visu
  al Studio¥2017¥Community

> %CMAKE_BIN% CMakeLists.txt -G "%GENERATOR%" -A x64 -D CMAKE_GENERA
  TOR_INSTANCE="%CMAKE_GENERATOR_INSTANCE%" -D CMAKE_GENERATOR="%CM
  AKE_GENERATOR%" -S %SOURCE_DIR% -B %BINARIES_DIR%

> %CMAKE_BIN% --build %BINARIES_DIR% --config debug

> %CMAKE_BIN% --build %BINARIES_DIR% --config release
```

コマンド実行後に、

<BASE_DIR>¥lib¥assimp-5.0.0¥BINARIES¥x64¥include¥assimp¥config.h

を

<BASE_DIR>¥lib¥assimp-5.0.0¥include¥assimp¥config.h

にコピーする。

ビルドが成功すると「assimp-5.0.0¥BINARIES¥x64¥code」フォルダ下の「Release」、「Debug」フォルダに、以下の assimp のライブラリファイルが作成される。

- Release
  - assimp-vc141-mt.lib
  - assimp-vc141-mt.dll
- Debug

- assimp-vc141-mtd.lib
- assimp-vc141-mtd.dll
- assimp-vc141-mtd.pdb

以下の 2 つのユーザー環境変数を設定する。

変数名	変数値
ASSIMP_INC_DIR	<BASE_DIR>¥lib¥assimp-5.0.0¥include
ASSIMP_LIB_DIR	<BASE_DIR>¥lib¥assimp-5.0.0¥BINARIES¥x64¥code¥Release

3.5.2. Autodesk FBX SDK

Autodesk_FBX_Review_Win_64bit.exe を用いて<FBX_SDK_DIR>に FBX SDK をインストールする。そして以下の 2 つのユーザー環境変数を設定する。Visual Studio のバージョンは適宜読み替えること。

変数名	変数値
FBX_SDK_INC_DIR	<FBX_SDK_DIR>¥include
FBX_SDK_LIB_DIR	<FBX_SDK_DIR>¥lib¥vs2017¥x64¥release

3.5.3. CGFormatExt4KVS

「CGFormatExt4KVS」フォルダにある「kvsmake_libs.vc.conf_template」を「kvsmake_libs.vc.conf」にリネームし、「FBX_SDK_DIR」と「ASSIMP_DIR」にそれぞれライブラリのパスを設定する。このパスは半角スペースを含むことがあるため” ”で括ること。

Assimp および FBX を利用しない場合は、kvsmake_libs.vc.conf ファイル中に定義された変数を以下のように編集する。

```
CGFORMATEXT4KVS_SUPPORT_FBXSDK = 0
CGFORMATEXT4KVS_SUPPORT_ASSIMP = 0
```

Tools コマンドプロンプトで「<BASE_DIR>¥lib¥CGFormatExt4KVS¥Lib」フォルダに移動し、以下のコマンドを実行する。

```
kvsmake lib
```

コマンド実行が成功すると、「<BASE_DIR>¥Lib¥CGFormatExt4KVS¥Lib」フォルダに「LibCGFormatExt4KVS.lib」が作成される。

環境変数「CGFORMAT_EXT4KVS_SHADER_DIR」に、
値「<BASE_DIR>¥lib¥CGFormatExt4KVS¥Lib」を設定する。

3.6. ML-PBVR と依存ライブラリ

3.6.1. Oculus SDK

「<BASE_DIR>¥lib」フォルダ内に「ovr_sdk_win_1.30.0_public」フォルダを作成する。ovr_sdk_win_1.30.0_public.zip を解凍して、その中にあるファイルとフォルダを全て「<BASE_DIR>¥lib¥ovr_sdk_win_1.30.0_public¥」フォルダ内にコピーする。

Visual Studio のバージョンは適宜読み替えること。

以下の 2 つのユーザー環境変数を設定する。

変数名	変数値
OCULUS_INC_DIR	<BASE_DIR>¥lib¥ovr_sdk_win_1.30.0_public¥LibOVR¥Include
OCULUS_LIB_DIR	<BASE_DIR>¥lib¥ovr_sdk_win_1.30.0_public¥LibOVR¥Lb¥Windows¥x64¥Release¥VS2017

3.6.2. Dear ImGui

imgui-1.79.zip を「imgui-1.79」フォルダに解凍し、それを「<BASE_DIR>¥lib」フォルダに移動する。imgui-1.79_cmakefiles.zip を「imgui-1.79」フォルダに解凍し、それを「<BASE_DIR>¥lib¥imgui-1.79」フォルダに移動する。

Tools コマンドプロンプトで「<BASE_DIR>¥lib¥imgui-1.79」フォルダに移動し、以下のコマンドを実行する。Visual Studio のバージョンは適宜読み替えること。

```
cmake -G "Visual Studio 15 2017" -A x64 .
cmake --build . --config release
copy Release¥libimgui.lib .
copy examples¥Release¥libimgui_impl_opengl3.lib .
```

コマンド実行後、「<BASE_DIR>¥lib¥imgui-1.79」フォルダに「libimgui.lib」と「libimgui_impl_opengl3.lib」が生成される。

3.6.3. CS-IS-PBVR

CS-IS-PBVR のソースコードを
github (https://github.com/CCSEPBVR/CS-IS-PBVR)
から「<BASE_DIR>にクローンする。

そして ¥CS-IS-PBVR¥QtClient¥qtpbvr.conf にあるコンフィグファイルを以下のように編集して、VR モードでビルドできるようにする。

```
#PBVR_MODE = CS
#PBVR_MODE = IS
PBVR_MODE = VR
```

3.6.3.1. プロジェクトの設定

プロジェクトファイル「<BASE_DIR>¥CS-IS-PBVR¥QtClient¥QtClient.pro」をQtCreator で開く。プロジェクトの設定画面において左サイドバーの"Buld & Run"の下に表示されている"Desktop Qt 5.9.9 MSVC2017 64bit" > “Build”をクリックし「ビルド設定」を表示する(図 3.6.3-1)。Visual Studio のバージョンは適宜読み替えること。

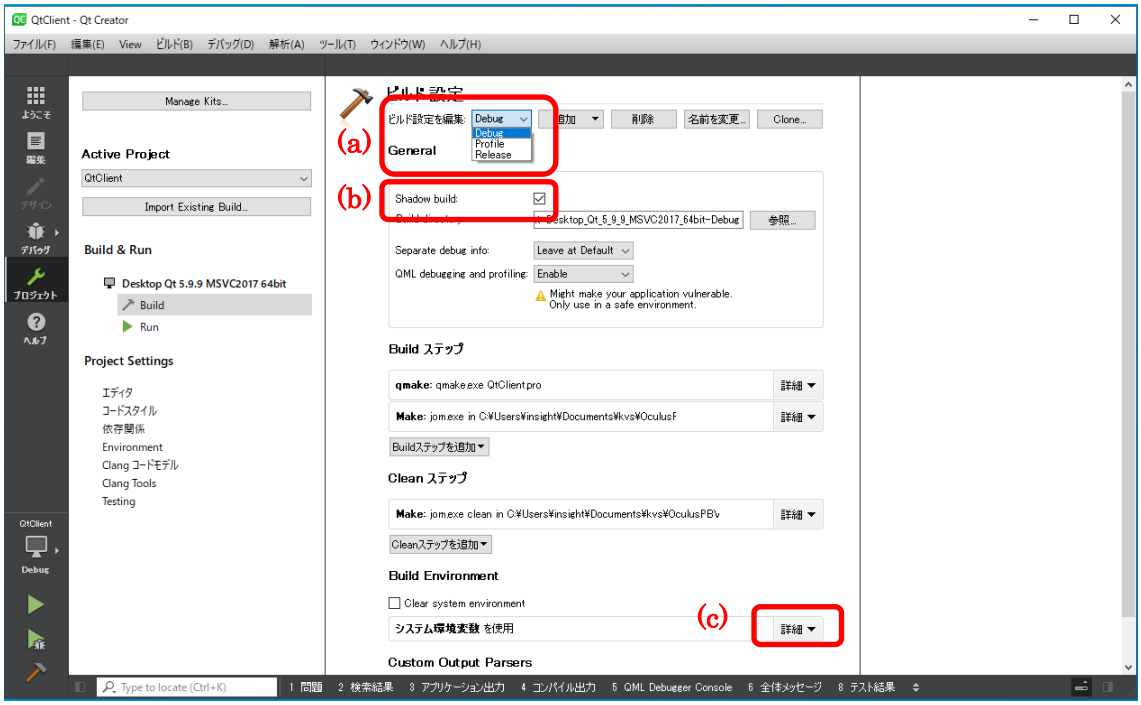


図 3.6.3-1 「Desktop Qt 5.9.9 MSVC2017 64bit」をクリックした直後の画面

(1) ビルド設定を編集

“Edit build configuration”(図 3.6.3-1 (a))において「Release」を選択する。

(2) General > Shadow Build

「General > Shadow Build」 (図 3.6.3-1 (b))をチェックする。

(3) Build Environment

“Build Environment”の「詳細」((図 3.6.3-1 (c)))をクリックし、以下の環境変数を追加する。

Variable (変数名)	Value (変数値)
GLEW_DIR	<BASE_DIR>¥lib¥glew-2.1.0

KVS_SOURCE	<BASE_DIR>¥KVS2.9mod4meta
IMGUI_DIR	<BASE_DIR>¥lib¥imgui-1.79

3.6.3.2. コンフィギュレーション

SETTINGS.pri ファイル内の DEFINES 変数の値で ML-PBVR の構成を設定できる。

(1) ミラー表示

ML-PBVR は HMD の画像をディスプレイ上の GUI にミラー表示する。DEFINES のデフォルト値は（ア）である。

(ア) DEFINES += MIRROR_IMAGE_BOTH_DISTORTED 両目画像(歪みあり)をミラー表示する。

(イ) DEFINES += MIRROR_IMAGE_BOTH
両目画像(歪みなし)をミラー表示する。

(ウ) DEFINES += MIRROR_IMAGE_LEFT_ONLY
左目画像のみをミラー表示する。

(エ) DEFINES += MIRROR_IMAGE_RIGHT_ONLY
右目画像のみをミラー表示する。

(2) オブジェクト操作

Touch コントローラは可視化オブジェクトを掴み、その動きでオブジェクトを座標変換する。その掴む動作は Touch コントローラの HandTrigger または IndexTrigger に紐付けられている。DEFINES のデフォルト値は（ア）である。

(ア) DEFINES += GRAB_USING_HAND_TRIGGER
中指ボタン(HandTrigger)に掴む動作を定義する。

(イ) DEFINES += GRAB_USING_INDEX_TRIGGER
人差し指ボタン(IndexTrigger)に掴む動作を定義する。

(3) デバッグ出力

標準出力に Scene, Screen, TouchController クラスの処理ログを出力する。

(ア) Scene クラス ：“DEFINES += DEBUG_SCENE”

(イ) Screen クラス ：“DEFINES += DEBUG_SCREEN”

(ウ) TouchController クラス ：“DEFINES += DEBUG_TOUCH_CONTROLLER”

3.6.3.3. ビルド

メニューバー ＞ 「ビルド」 ＞ 「プロジェクト “QtClient”をビルド」

3.7. サーバプログラム

CS-PBVR は大規模データを効率よく並列処理するために領域分割するフィルタプログラムと、サーバ上でボリュームデータを可視化用粒子に変換する Particle Sampler、そしてユーザ PC 上でボリュームレンダリングする Particle Renderer で構成されている。ML-PBVR は CS-PBVR の Particle Renderer に関する VR 拡張であり、フィルタプログラムや Particle Sampler は CS-PBVR と共通である。詳細は

CS-PBVR のマニュアル（https://ccse.jaea.go.jp/software/PBVR/）を参照すること。

4. ビルド手順

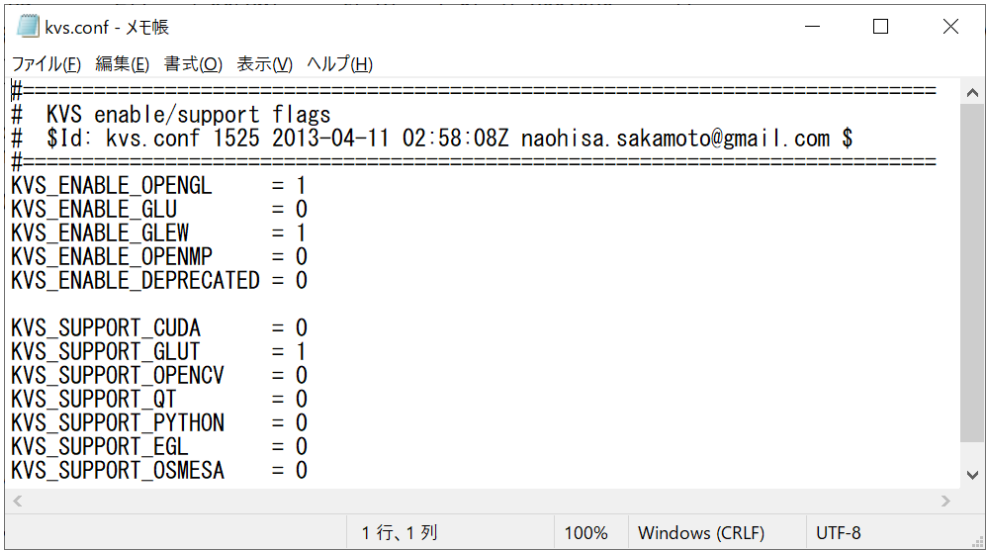
ML-PBVR のビルド手順を本節にまとめる。

x64 Native Tools Command Prompt for VS 2019 を起動する。

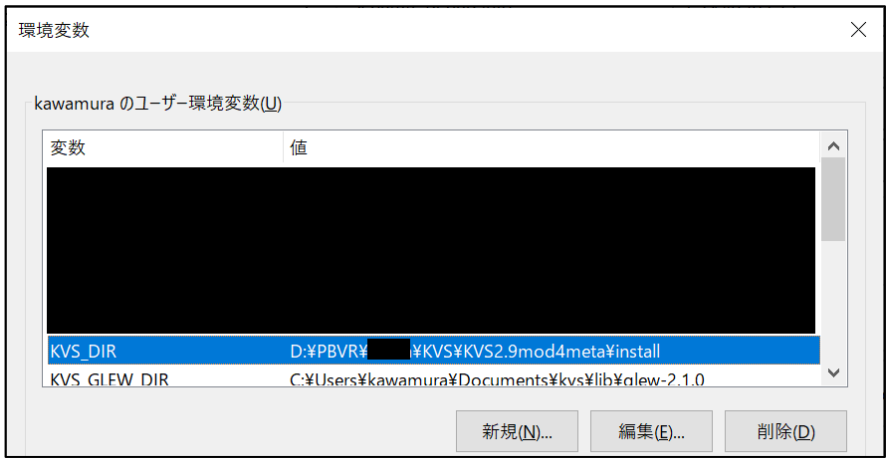
CCSE の git からクライアントプログラム、KVS ライブラリのソースコードをクローンする。

```
git clone git@github.com:CCSEPBVR/CS-IS-PBVR.git
cd CS-IS-PBVR
git checkout release_cs_is_pbvr_v2.2.1
cd ..
git clone git@github.com:CCSEPBVR/KVS.git
```

D:¥WorkDIR¥KVS¥KVS2.9mod4meta¥kvs.conf を図の通りに編集する。



環境変数 KVS_DIR を設定する。



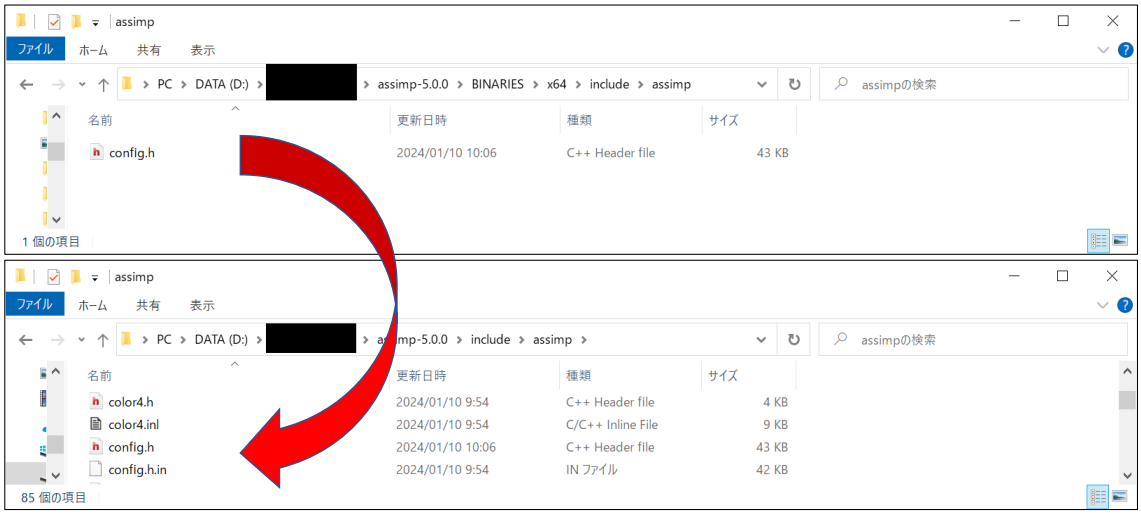
KVS ライブラリをビルド、インストールする。

```
  nmake
  nmake install
```

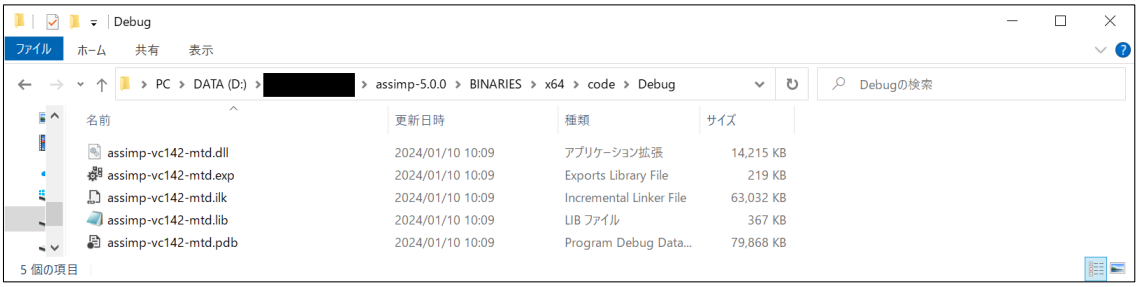
Assimp ライブラリをダウンロード、インストールする。

```
> SET SOURCE_DIR=.

> SET GENERATOR=Visual Studio 16 2019

> SET BINARIES_DIR="./BINARIES/x64 “

> SET CMAKE_GENERATOR_INSTANCE=C:¥Program Files (x86)¥Microsoft Visu
al Studio¥2019¥Community

> cmake CMakeLists.txt -G "%GENERATOR%" -A x64 -D CMAKE_GENERATOR_IN
> STANCE="%CMAKE_GENERATOR_INSTANCE%" -D CMAKE_GENERATOR="%CMAKE_GE
NERATOR%" -S %SOURCE_DIR% -B %BINARIES_DIR%

> cmake --build %BINARIES_DIR% --config debug

> cmake --build %BINARIES_DIR% --config release
```

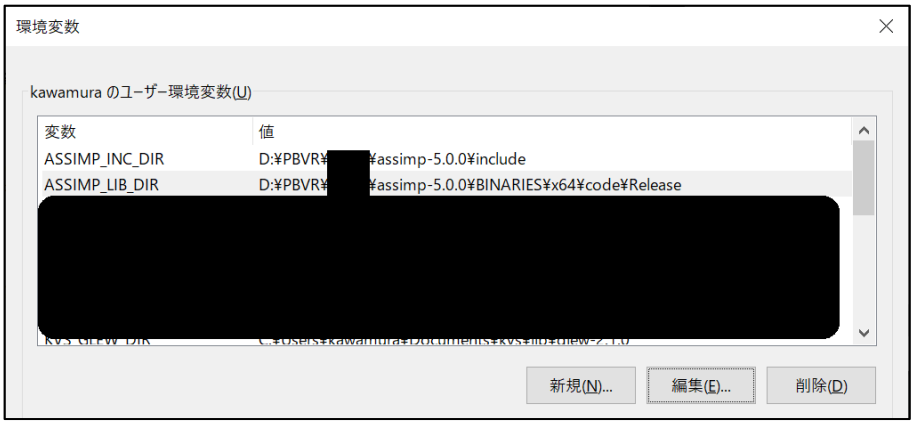
Assimp ライブラリを
D:¥WorkDIR¥assimp-5.0.0¥BINARIES¥x64¥include¥assimp¥config.h
から
D:¥WorkDIR¥assimp-5.0.0¥include¥assimp
に移動する。



D:¥WorkDIR¥assimp-5.0.0¥BINARIES¥x64¥code¥Releasee　と
D:¥WorkDIR¥assimp-5.0.0¥BINARIES¥x64¥code¥Debug
にライブラリファイルが生成されているか確認する。

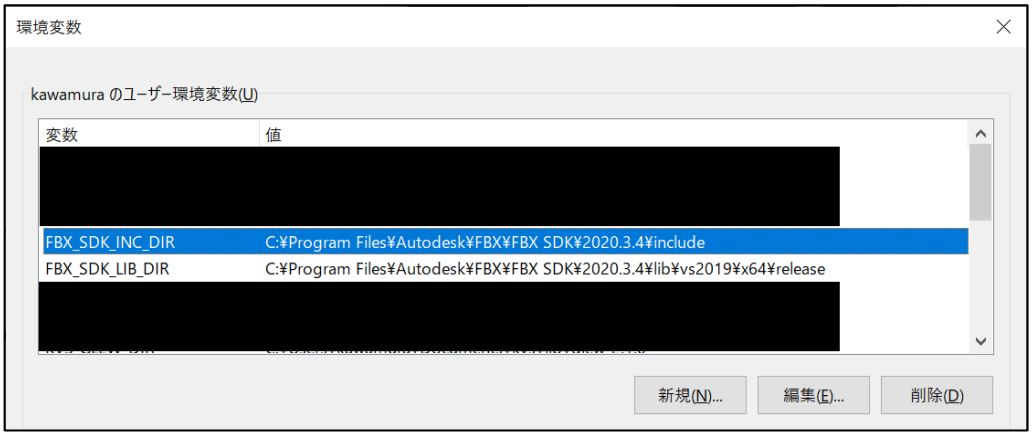


環境変数 ASSIMP_INC_DIR と ASSIMP_LIB_DIR を設定する。



FBX SDK をダウンロードし、インストーラの指示に従いインストールする。本手順ではインストール先のフォルダは初期設定を使用する。

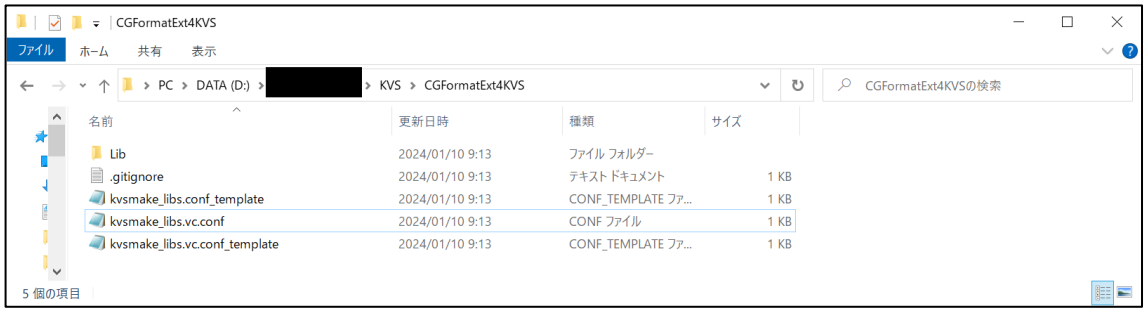
環境変数 FBX_SDK_INC_DIR と FBX_SDK_LIB_DIR を設定する。



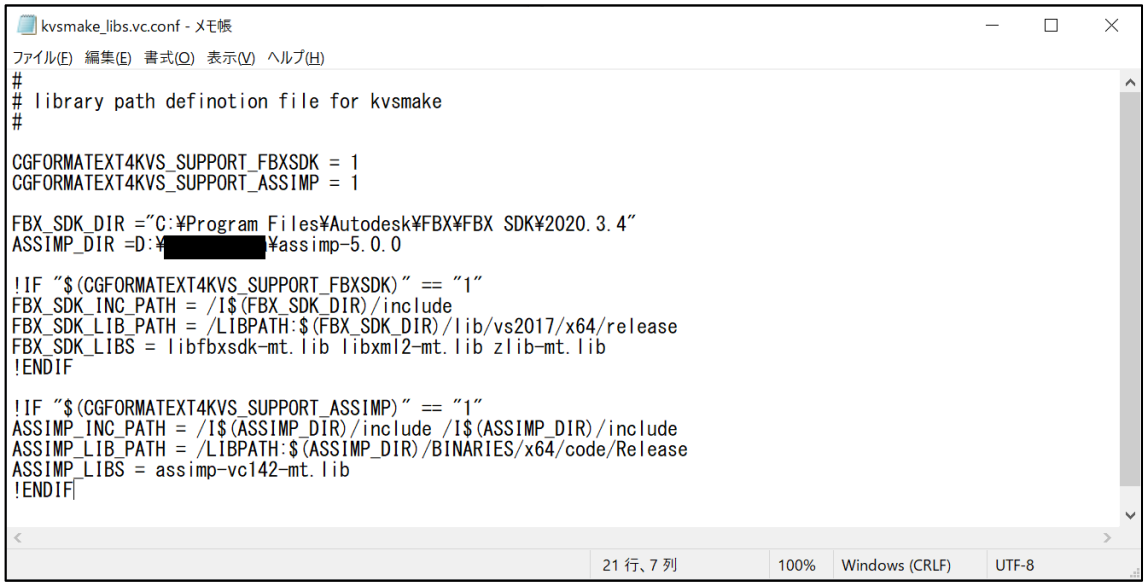
CGFormatExt4KVS をビルドするために、

D:¥WorkDIR¥KVS¥CGFormatExt4KVS¥kvsmake_libs.vc.conf_template

をコピーして kvsmake_libs.vc.conf にリネームする。



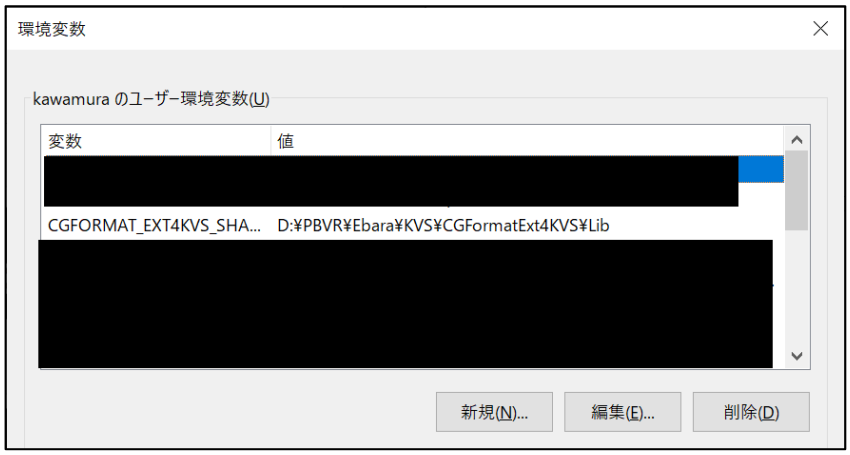
kvsmake_libs.vc.conf を編集し、ASSIMP と CGFormatExt4KVS の設定を追加する。



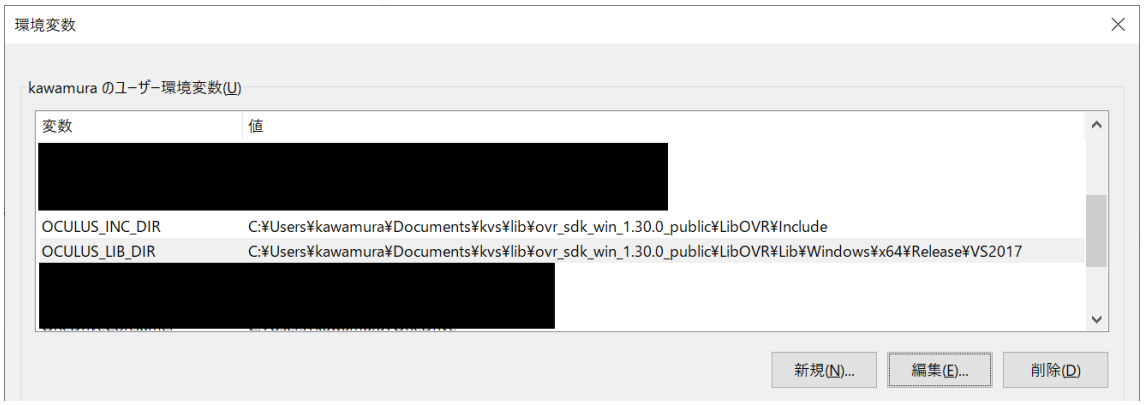
CGFormatExt4KVS をビルドする。

```
kvsmake lib
```

環境変数 CGFORMAT_EXT4KVS_SHADER_DIR を設定する。



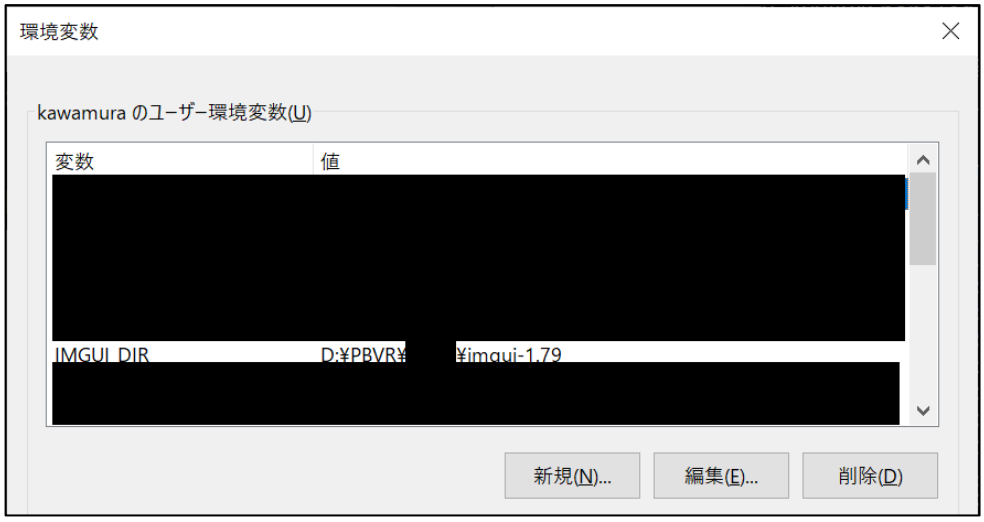
Oculus SDK をダウンロードし、環境変数 OCULUS_INC_DIR と OCULUS_LIB_DIR を設定する。必ず v1.30.0 を使用すること。v32.0.0 を使用してクライアントをビルドすると Cannot initialize LibOVR というエラーが出力される。



Dear ImGui をダウンロードしインストールする。

```
cmake -G "Visual Studio 16 2019" -A x64 .
cmake --build . --config release
copy Release¥libimgui.lib .¥
copy examples¥Release¥libimgui_impl_opengl3.lib .¥
```

環境変数 IMGUI_DIR を設定する。



QtCreator 上で D:¥WorkDIR¥CS-IS-PBVR¥QtClient¥QtClient.pro を開き、App.pro 60,61 行目を図の通り編集する。

```
58 ## for VR/MR mode (windows only)
59 win32{
60 #    contains(DEFINES, VR_MODE)|contains(DEFINES, MR_MODE) {
61     contains(DEFINES, VR_MODE)|contains(DEFINES, MR_MODE) | contains(DEFINES, IS_MODE) {
62         LIBS += -L$$(OCULUS_LIB_DIR) -llibOVR                    You 2023-12-01
63         LIBS += -L$$(IMGUI_DIR) -llibimgui -llibimgui_impl_opengl3
64         #-llibimgui_impl_win32
65
66         contains(DEFINES, MR_MODE) {
67             LIBS += -L$$(CUDA_PATH)/lib/x64 -lcuda -lcudart_static -lcublas
68             LIBS += -L$$(ZED_SDK_ROOT_DIR)/lib -lsl_zed64
69         }
70     }
```

commit caef4680a13bc74a7f635ffdeab
Author: unknown <kawamura@precisi
Date: 2023-12-01 11:38:31

release version 2.2.1

QtCreator 上で D:¥WorkDIR¥CS-IS-PBVR¥QtClient¥QtClient.pro を開き、SETTINGS.pri の 38,46,48,149,150 行目を下図の通りに編集する。

```
35 contains(DEFINES, VR_MODE)|contains(DEFINES, MR_MODE) {
36     DEFINES += DISP_MODE_VR
37 } else {
38     DEFINES += DISP_MODE_VR
39 }
```

```
45 DEFINES += CGFORMATEXT4KVS_SUPPORT_FBXSDK
46 DEFINES += ENABLE_FBXSDK
47 contains(DEFINES, CS_MODE){
48 #DEFINES += ENABLE_LIBLAS
49 }
```

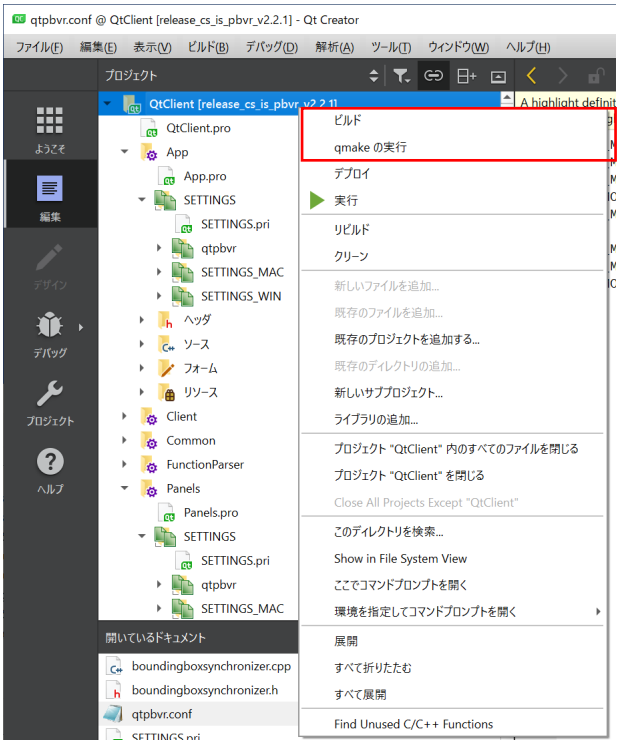
```
148 INCLUDEPATH += $$(CGFORMAT_EXT4KVS_SHADER_DIR)
149 #contains(DEFINES, VR_MODE)|contains(DEFINES, MR_MODE) {
150 contains(DEFINES, VR_MODE)|contains(DEFINES, MR_MODE) | contains(DEFINES, IS_MODE) {
151     INCLUDEPATH += $$(OCULUS_INC_DIR)
152
153     contains(DEFINES, MR_MODE) {
154         INCLUDEPATH += '$$(ZED_SDK_ROOT_DIR)/include'
155         INCLUDEPATH += '$$(CUDA_PATH_V11_0)/include'
156     }
157 }                                                    You 2023-12-01
```

qtpbvr.conf を下図の通りに編集する。

A highlight definition was not found for this file. Would you like to download additional highlight definition files? 定義をダウンロード 今後このメッセージを表示しない

```
#PBVR_MODE - CS (ClientServer), or IS (Insitu) or VR (Virtual Reality) or MR (Mixed Reality) - Needed on all platforms
#PBVR_MODE = CS
PBVR_MODE = IS
#PBVR_MODE = VR
#PBVR_MODE = MR

#REND_MODE - CPU, or GPU - Needed on all platforms
#REND_MODE = CPU
REND_MODE = GPU


```

QtCreator 上で qmake の実行、ビルドを行う。



5. ML-PBVR の起動

ML-PBVR は IS-PBVR と共通のサーバを利用する。ML-PBVR のユーザはホストとゲストに分類される。ホストは解析の主体であり可視化パラメータを指定する役割であり、ゲストはその可視化結果を受信する。概略を図 3.6.3-1 に示す。

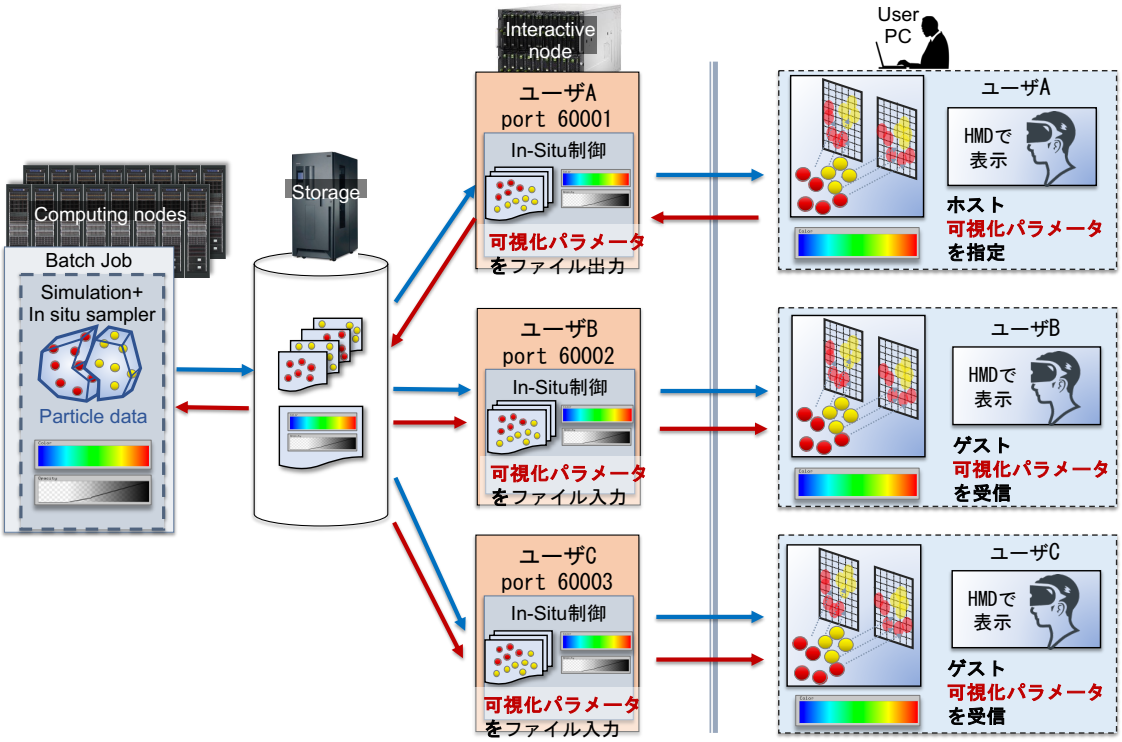


図 3.6.3-1 ML-PBVR の概略

本節は ML-PBVR の Particle Renderer の起動方法を記述する。サーバの利用方法は IS-PBVR のマニュアルを参照すること。ML-PBVR はプロジェクトファイルから起動される。"Buld & Run" > "Desktop Qt 5.9.9 MSVC2017 64bit" > “Run” から開かれる「実行時の設定」画面からコマンドライン引数が入力できる。Visual Studio のバージョンは適宜読み替えること。ML-PBVR は左サイドバー下のボタン、もしくはメニューバーから実行される。コマンドライン引数として以下のオプションが利用可能である。

- `-host` (ホストとして実行する)
- `-vin` (可視化データファイル名を指定する)
- `-tf` (伝達関数ファイル名を指定する)
- `-lefty` (左利き用設定で実行する)
- `-cgmodel <FBX file or 3ds file>` (FBX ファイルもしくは 3ds ファイルを表示する)

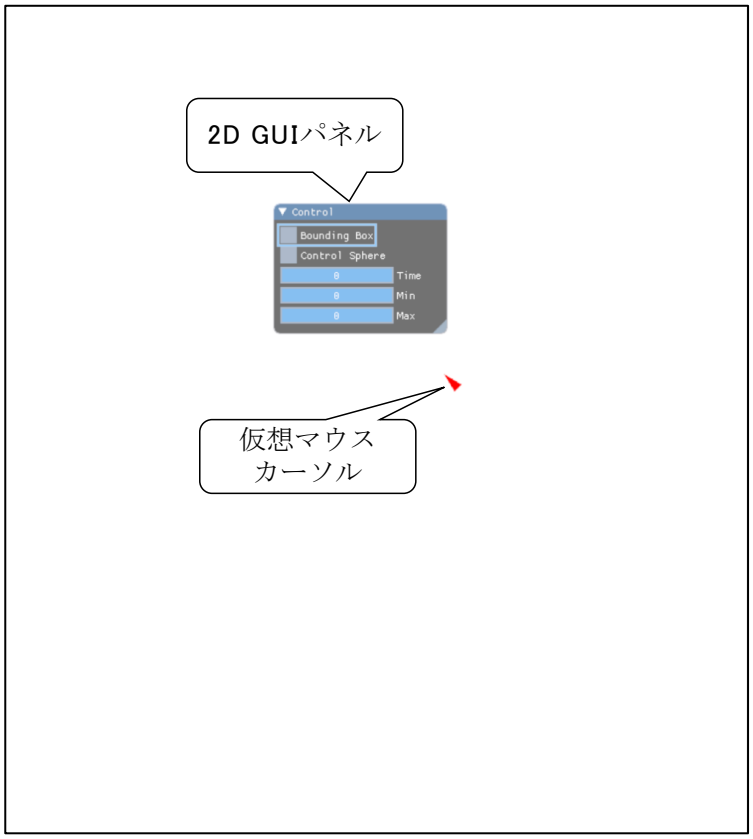


図 3.6.3-2 起動直後の制御パネルとマウスカーソル

実行直後の 3D 空間内には、レイオーバーで表示される制御パネルとそれを操作するための仮想マウスカーソルが描画される(図 3.6.3-2)。

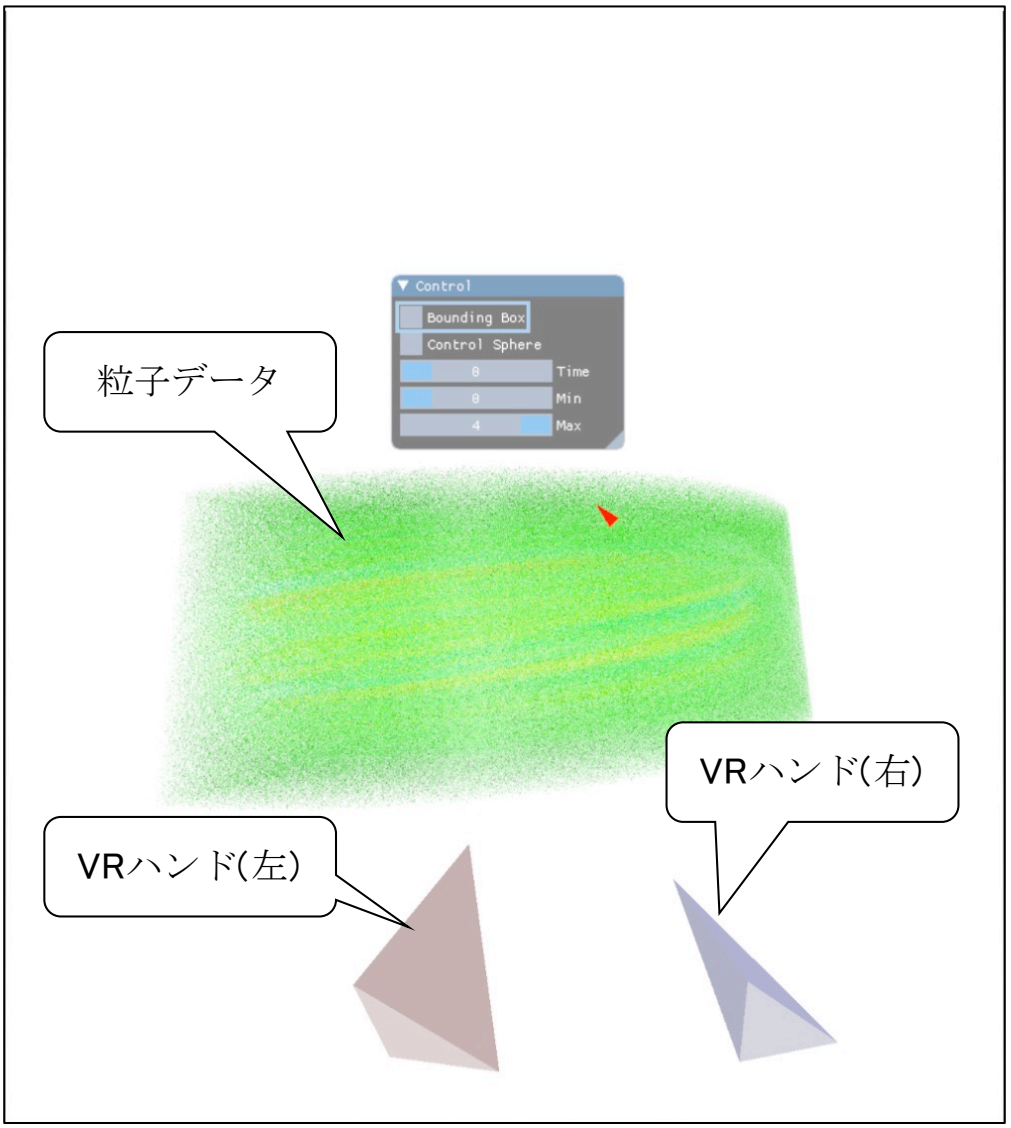


図 3.6.3-3 可視化データと VR ハンド

レンダラーはサーバから粒子データを受信し、可視化データと VR ハンドを描画する(図 3.6.3-3)。

6. VR 空間での操作

6.1. 制御パネル

制御パネルは赤い三角形のマウスカーソルで操作できる。右利きモードの場合、マウスカーソルは Touch Controller の左スティックで移動され、A ボタンでクリックされる(図 3.6.3-1)。左利きモードの場合、マウスカーソルは右スティックで移動され、X ボタンでクリックされる(図 3.6.3-2)。

図 3.6.3-1 右利きモード("-lefy"オプションなし)のマウスカーソルの操作

図 3.6.3-2 左利きモード(-lefty"オプションあり)のマウスカーソルの操作

制御パネル（図 3.6.3-3）は以下の機能を提供する。

(1) バウンディングボックスの表示 On/Off の切り替え（図 3.6.3-4）

(2) 制御球の表示 On/Off の切り替え（図 3.6.3-5）

(3) 時間ステップの設定

(a) 表示ステップの指定

(b) 最小ステップの指定

(c) 最大ステップの指定

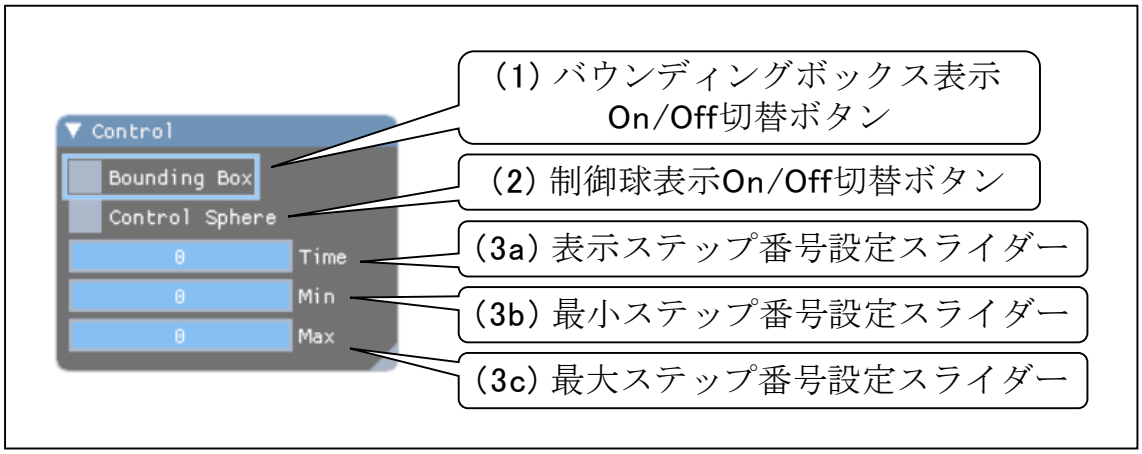


図 3.6.3-3 GUI パネルの操作項目

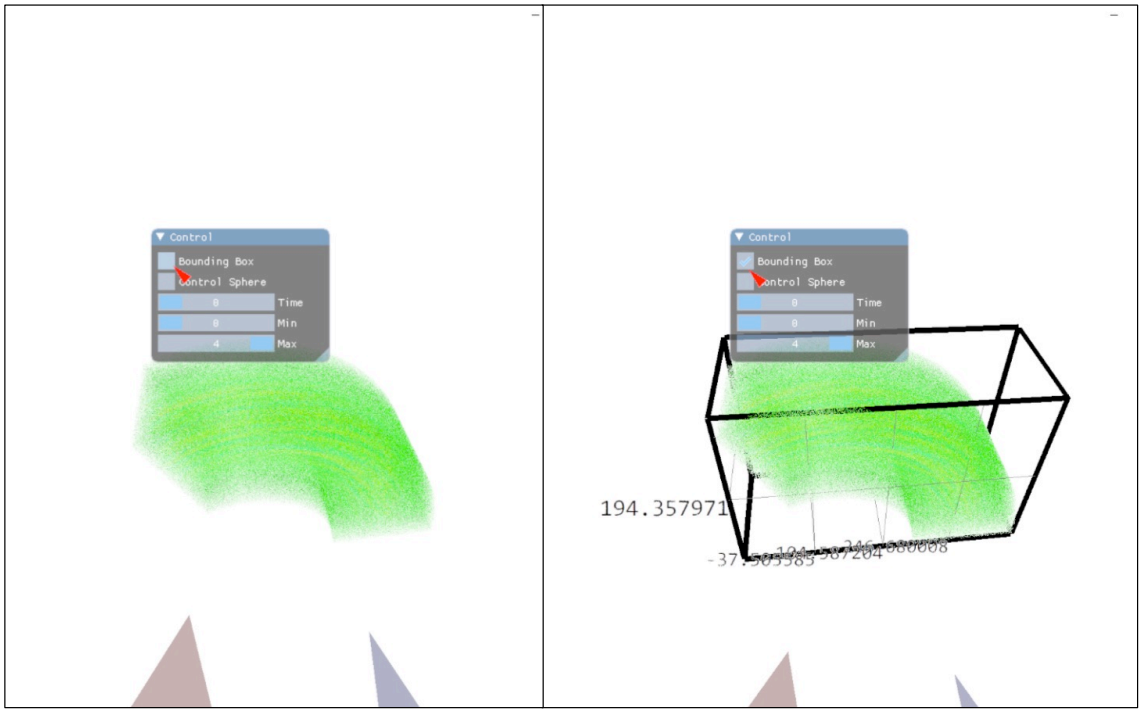


図 3.6.3-4 バウンディングボックス表示 Off(左)と表示 On(右)

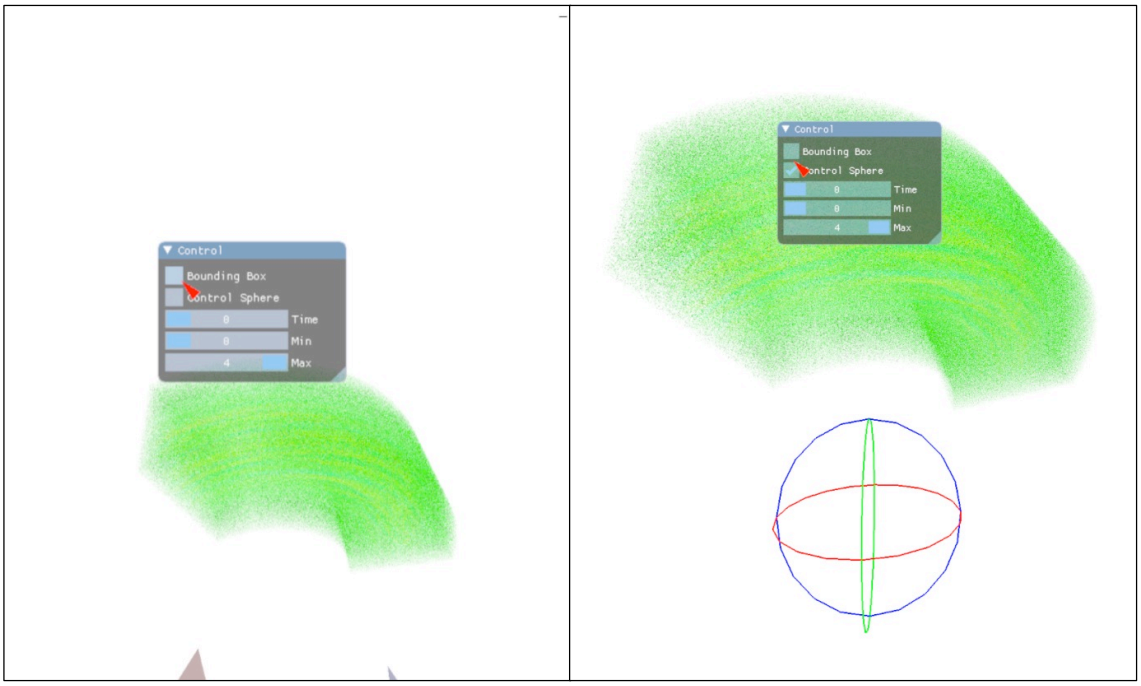


図 3.6.3-5 制御球表示 Off(左)と表示 On(右)

時間ステップについて、一番上のスライダーが表示ステップの設定、次が表示可能なステップの下限、一番下が表示可能なステップの上限である。制御パネルの時間ステップとディスプレイ上の GUI は連動している(図 3.6.3-6)。

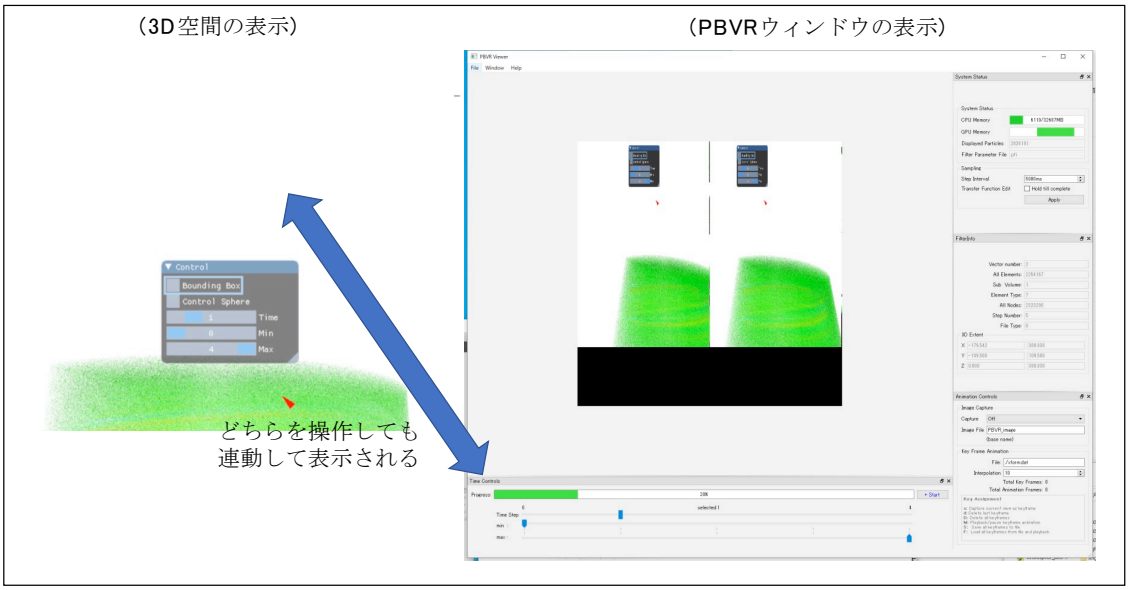


図 3.6.3-6 VR 空間とディスプレイ上の GUI

6.2.　座標変換

オブジェクトは Touch コントローラーのトリガーボタンで掴まれ、ジェスチャーで座標変換される。トリガーボタンは初期状態で HandTrigger (図 3.6.3-1)である。

図 3.6.3-1 トリガーボタンの位置

両手を同じ方向に動かすジェスチャーでオブジェクトが並行移動する(図 3.6.3-2)。

図 3.6.3-2 平行移動

両手を離す、あるいは近づけるジェスチャーでオブジェクトが拡大あるいは縮小する(図 3.6.3-3)。

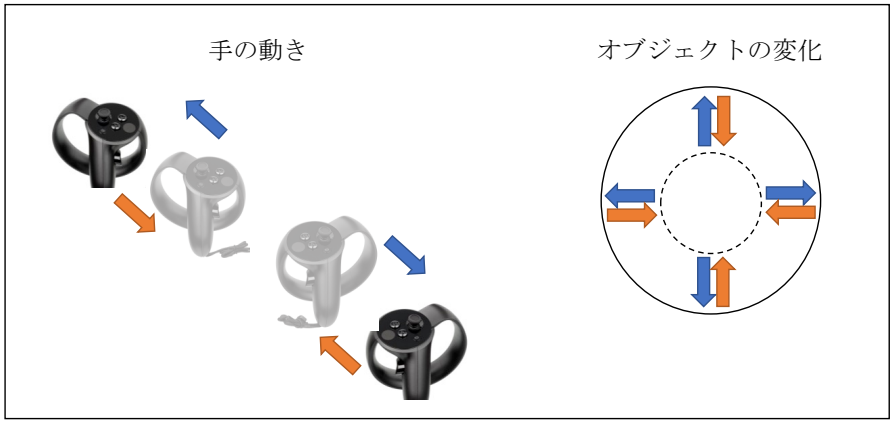


図 3.6.3-3 拡大縮小

両手を回すジェスチャーでオブジェクトが回転する（図 3.6.3-4）。

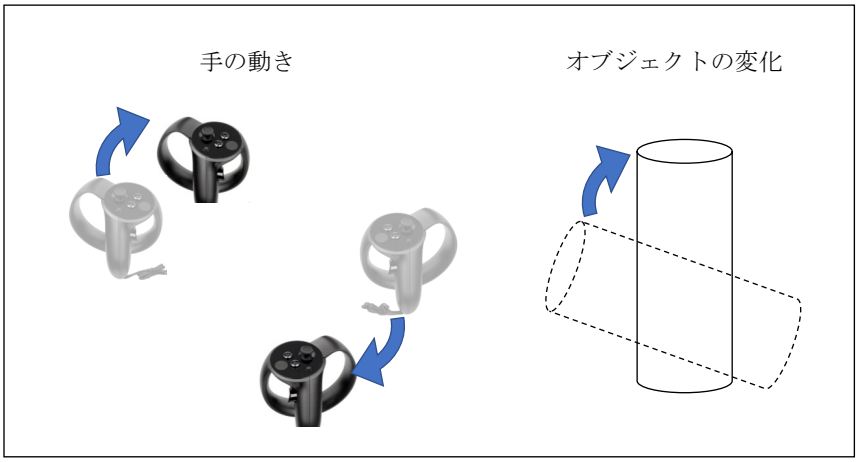


図 3.6.3-4 回転の操作方法

7. ML-PBVR の終了

7.1. クライアントの終了

クライアントはディスプレイ上のメインウィンドウの左上の×ボタンで終了する。サーバはターミナル上から Ctrl+C を入力して終了する。