

Multi-point Remote VR Visualization Application Multi- Location-PBVR

User Manual

Ver. 1.0

Center for Computational Science & e-System, Japan Atomic Energy Agency
2024/02/29

Table of contents

Release history	3
1. Introduction.....	4
1.1. Environment	5
2. Dependency Library.....	5
2.1. KVS2.9mod4meta	5
2.2. CGFormatExt4KVS.zip	6
2.3. assimp-5.0.0.zip.....	6
2.4. fbx20195_fbxsdk_vs****_win.exe	6
2.5. freeglut-MSVC-3.0.0-2.mp.....	6
2.6. glew-2.1.0-win32	6
2.7. ovr_sdk_win_1.30.0_public.....	6
2.8. cmake-3.16.3-win64-x64	6
2.9. imgui-1.79.....	6
3. Setup.....	7
3.1. Visual C++ Setting.....	7
3.2. Qt Creator Setting	9
3.3. CMake	11
3.4. KVS2.9mod4meta and Dependency.....	11
3.4.1. FreeGLUT	11
3.4.2. GLEW	11
3.4.3. KVS2.9mod4meta	11
3.5. CGFormatExt4KVS and Dependency	12
3.5.1. Assimp	12
3.5.2. Autodesk FBX SDK.....	13
3.5.3. CGFormatExt4KVS.....	13
3.6. ML-PBVR and Dependency.....	14
3.6.1. Oculus SDK.....	14
3.6.2. Dear ImGui	14
3.6.3. CS-IS-PBVR.....	14
3.7. Server Program.....	17
4. Build procedure	18
5. Launching ML-PBVR.....	25
6. Operation in VR space	28
6.1. Control Panel	28
6.2. Coordinate Transformation	31

7. OculusPBVR_JAEA の終了	32
------------------------------	----

Release history

version	update	contents
1.0	2023/03/15	First version

1. Introduction

This manual describes how to install and use Multiple Location PBVR (ML-PBVR), a multi-location VR visualization application developed at Center for Computational Science & e-Systems (CCSE), Japan Atomic Energy Agency (JAEA). ML-PBVR can visualize a large-scale simulation on a remote supercomputer on a PC at the same time as it is computed. ML-PBVR is capable of VR visualization using a head-mounted display (HMD), and allows interactive analysis among users by allowing multiple users to change visualization parameters.

Since the computation speed of today's supercomputers far exceeds the I/O speed, output of computation results has become difficult. As a result, conventional visualization methods that transfer calculation results on remote storage to a PC at hand for visualization have become difficult to apply. In-Situ visualization avoids large data I/O by having the visualization program coupled to the simulation execute the visualization process simultaneously with the calculation, and can reliably generate visualized images. Interactive in-situ visualization has been difficult with conventional polygon-based methods. CCSE has developed an In-Situ visualization framework, IS-PBVR, which enables interactive visualization using PBVR (Particle Based Volume Rendering) and interactive In-Situ control technology. PBVR is a particle-based visualization method developed in Koyamada Laboratory, Kyoto University. ML-PBVR is an extension of IS-PBVR for multi-location visualization. ML-PBVR consists of three components: a particle sampler, a daemon, and a PBVR client. ML-PBVR allows multiple users to share the same visualization data via files on storage by invoking the daemon respectively.

(1) Particle Sampler

The particle sampler is a visualization library that is coupled to a simulation code to generate particles in the same environment as the calculation. To build a simulation + in-situ visualization code, an array of calculation results is passed to the visualization functions provided by the particle sampler and inserted into the simulation code. The particle sampler refers to the visualization parameter file on the storage, converts the calculation results of each time step into compressed particles for visualization, and outputs them on the storage. The particle files are output from each process in a distributed manner.

(2) Daemon

The daemon runs on the login node or the interactive job and mediates between the particle files on the storage and the visualization parameters sent from the PBVR client. The daemon monitors the files on the storage, aggregates the distributed output particle

files, and transfers them to the user PC via the network. The daemon also receives the visualization parameters sent from the PBVR client and outputs them as a visualization parameter file to be referenced by the particle sampler. Since this operation is executed asynchronously with the simulation, interactive control is possible without interfering with the simulation.

(3) PBVR Client

The PBVR Client runs on the user's PC and provides a viewer to display the visualization results and a GUI to edit the visualization parameters. PBVR Client communicates with the daemon using port forwarding. The PBVR client receives the particle data from the daemon, displays the volume rendered image on the viewer, and sends the visualization parameters edited by the user on the GUI to the daemon.

1.1. Environment

ML-PBVR has been tested in the following environments

	Specification
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
Compiler	Visual Studio 2017 / Visual Studio 2019
IDE	Qt Creator 5.9 / Qt Creator 6.5

2. Dependency Library

The following libraries are required to build and run ML-PBVR. Those of KVS2.9mod4meta and CGFormatExt4KVS are available from following link.

github(<https://github.com/CCSEPBVR/KVS>)

The others are available from links described in ReadMe.txt of above github.

2.1. KVS2.9mod4meta

KVS2.9mod4meta is a KVS modified for ML-PBVR and provides basic visualization capabilities for HMD Meta.

2.2. CGFormatExt4KVS.zip

CGFormatExt4KVS is a library that converts polygon data in 3DS and FBX formats into polygon data in VR-KVS format.

2.3. assimp-5.0.0.zip

Assimp is a C++ library that provides the ability to load polygon data in 3DS format. It is statically linked at CGFormatExt4KVS build time, so it is not needed at runtime.

2.4. fbx20195_fbxsdk_vs****_win.exe

This is an installer for FBX SDK, a library provided by Autodesk that provides the ability to read polygon data in FBX format, which is required to build and run CGFormatExt4KVS and ML-PBVR. The version of Visual Studio is replaced as needed.

2.5. freeglut-MSVC-3.0.0-2.mp

This is a binary package of OpenGLUT, a GLUT for Windows, required to build and run VR-KVS and ML-PBVR.

2.6. glew-2.1.0-win32

This is a GLEW binary package for Windows and is required to build and run VR-KVS and ML-PBVR.

2.7. ovr_sdk_win_1.30.0_public

Oculus SDK is required to build ML-PBVR.

2.8. cmake-3.16.3-win64-x64

Assimp build requires cmake 3.16 or higher.

2.9. imgui-1.79

Dear imgui is an open source C++ library that generates a 2D GUI on the HMD and is required to build ML-PBVR.

3. Setup

This chapter describes the pre-configuration and procedures for compiling ML-PBVR, as well as the pre-configuration of the filter and server programs required to run ML-PBVR. These operations need only be performed once for the first time.

In the following, the directory where the work starts is denoted as <BASE_DIR>.

3.1. Visual C++ Setting

To begin, download and install Visual C++ 2017/2019 (Community Edition or higher required) from the Microsoft website (<https://visualstudio.microsoft.com/ja/vs/older-downloads/>). Next, start Visual Studio Installer and select "Advanced ▼" -> "Change" (Figure 3.4.1-1).

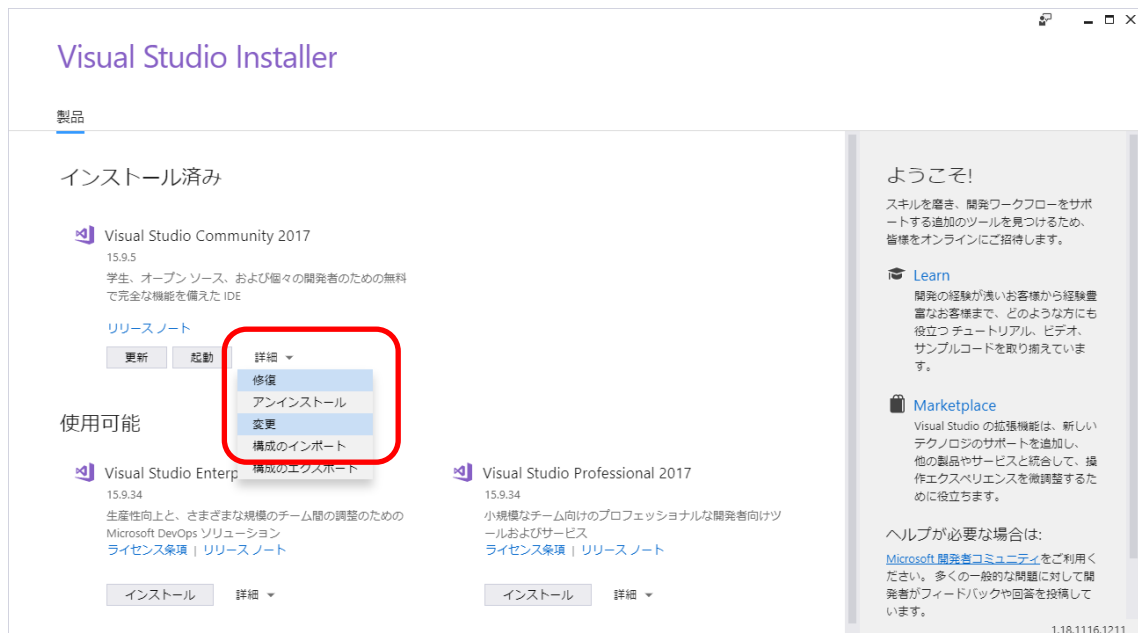


Figure 3.4.1-1 Advanced ▼ -> Change

On the component selection screen, install the following components (minor versions may be different).

- (A) Select "Workload" tab
 - (1) Select "Desktop Development with C++" (0)
- (B) Select "Individual components" tab
 - (1) Windows 10 SDK (10.0.17763.0) (Figure 3.4.1-3)
 - (2) Windows 10 SDK for Desktop C++(10.0.16299.0)[x86 or x64] (Figure 3.4.1-3)
 - (3) Visual C++ tools for CMake(Figure 3.4.1-4)

(4) VC++ 2017/2019 (Figure 3.4.1-4)

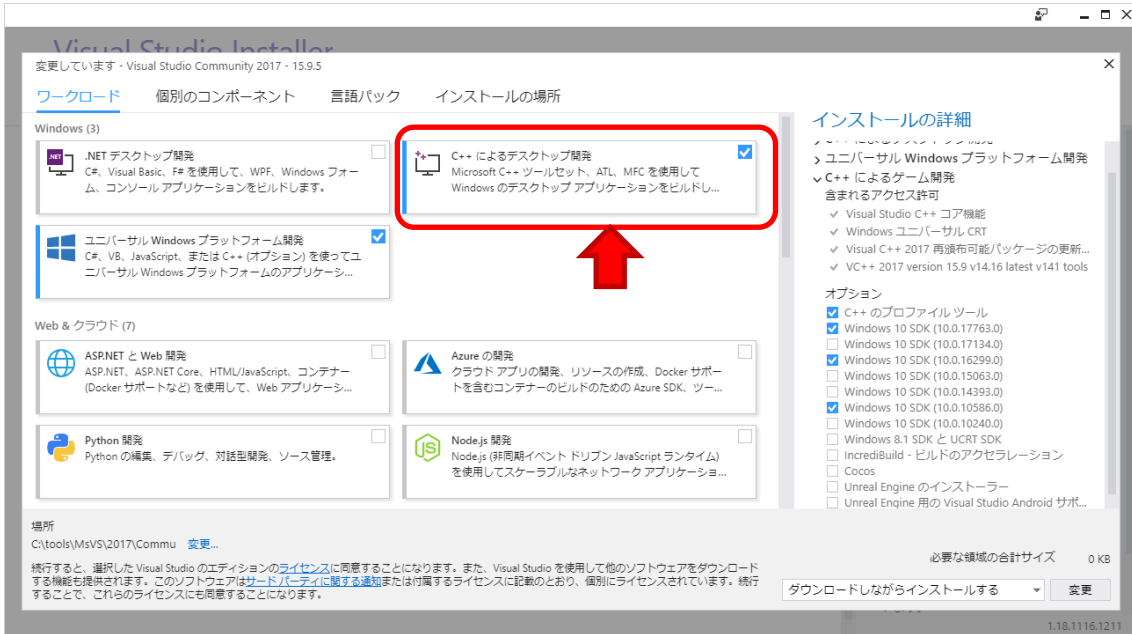


Figure 3.4.1-2 Desktop Development with C++

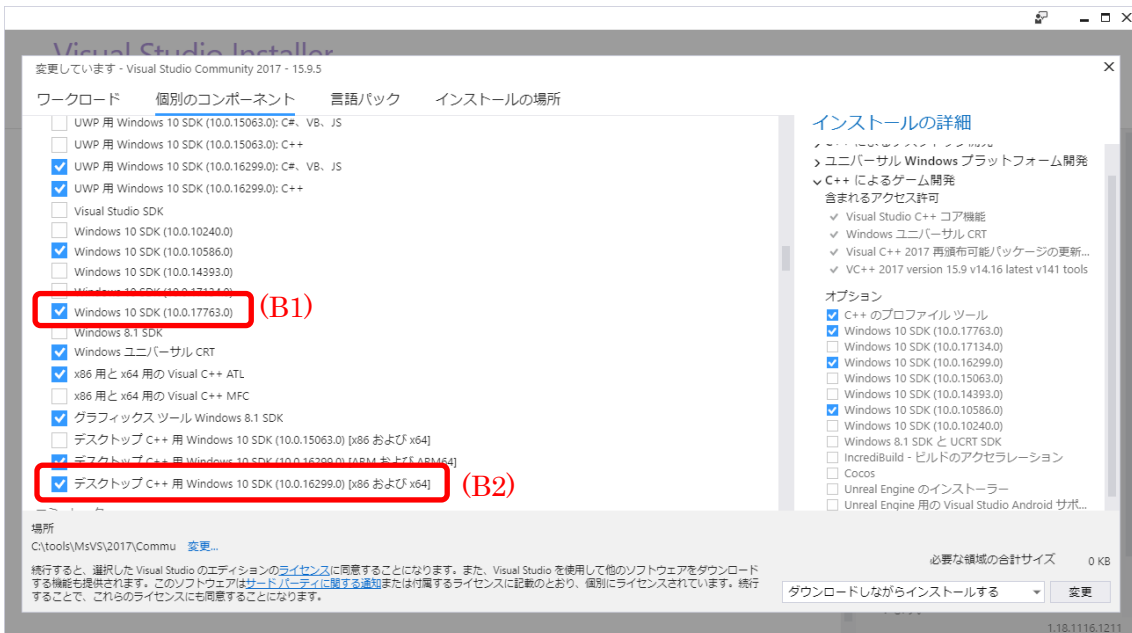


Figure 3.4.1-3 Individual components - Windows SDK

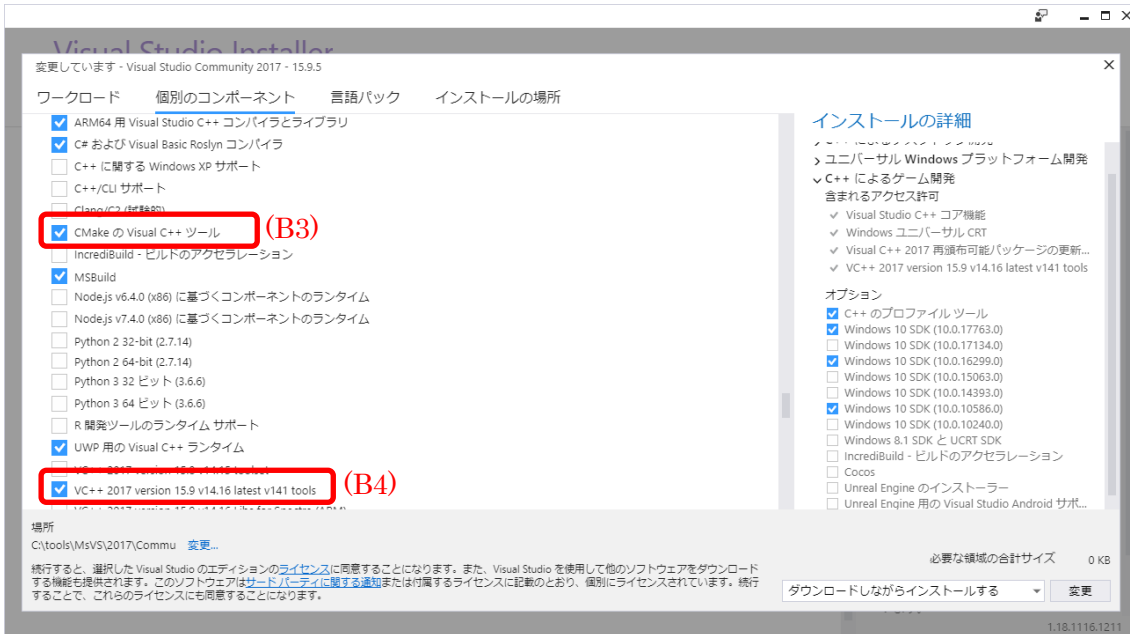


Figure 3.4.1-4 Visual C++ tools for CMake and VC++ tools version 15.9

After selecting components, click the "Change" button in the lower right corner to install.

3.2. Qt Creator Setting

Download and run the Qt Open Source installer (Qt Online Installer <https://qt.io/download-open-source>) from the official Qt website.

In the Select categories, uncheck "latest release" and "Preview" and leave only "LTS" checked (Figure 3.4.1-1).

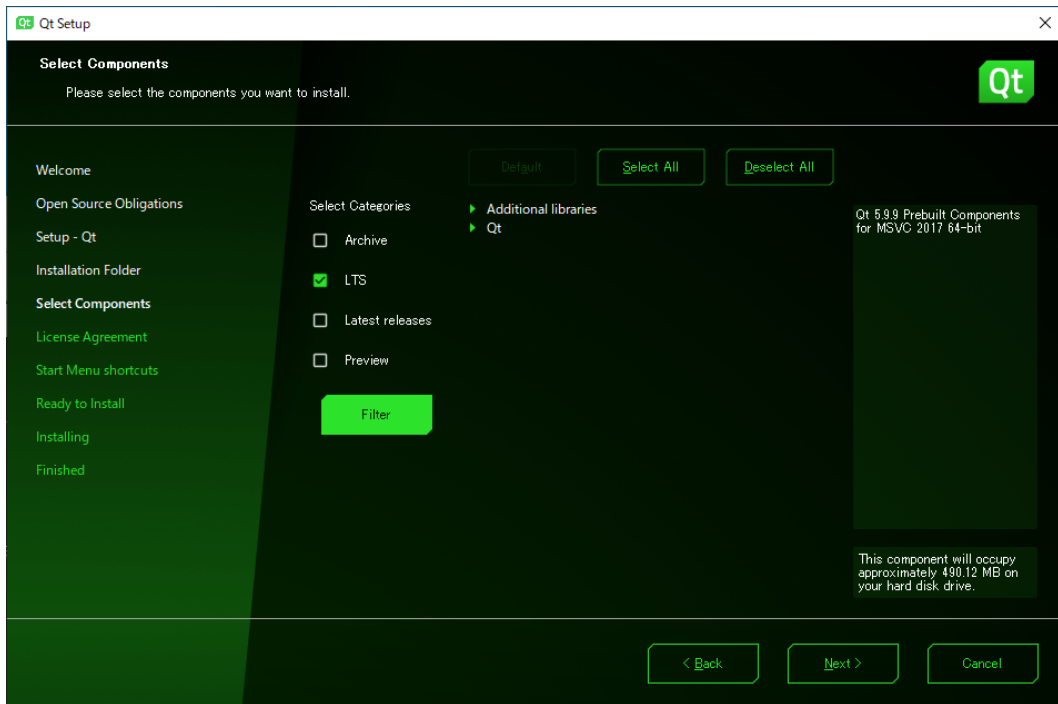


Figure 3.4.1-1 Qt installer, “Select category”

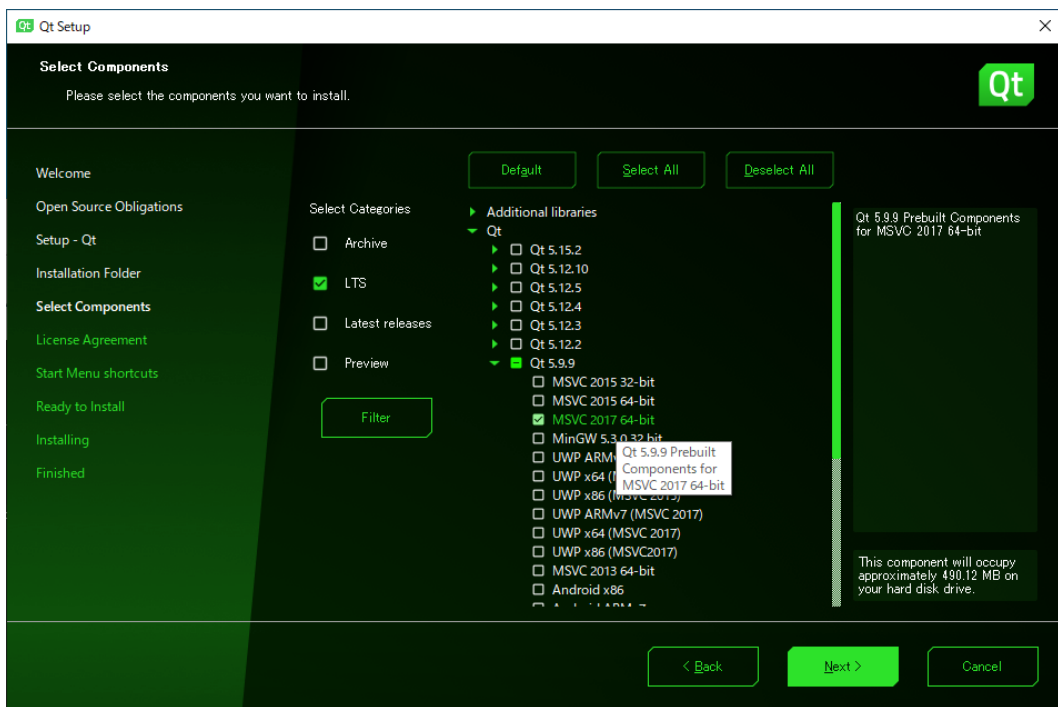


Figure 3.4.1-2 Qt installer, components

Then click "► Qt" and select a component from the table below (Figure 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 Debugger 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

Decompress `cmake-3.16.3-win64-x64.zip` into the "`cmake-3.16.3-win64-x64`" folder and move it into the "`<BASE_DIR>\tools`" folder.

3.4. KVS2.9mod4meta and Dependency

3.4.1. FreeGLUT

Decompress the "`freeglut-MSVC-3.0.0-2-mp.zip`" into the "`freeglut`" folder and move it into the "`<BASE_DIR>\lib`" folder. Copy "`freeglut.lib`" in the "`freeglut\lib\x64`" folder to the "`<BASE_DIR>\lib`" folder.

Set the user environment variable "`KVS_GLUT_DIR`" to the value "`<BASE_DIR>\lib\freeglut`".

3.4.2. GLEW

Decompress `glew-2.1.0-win32.zip` into the "`glew-2.1.0`" folder and move it to the "`<BASE_DIR>\lib`" folder. Copy "`glew32.lib`" and "`glew32s.lib`" in the "`glew-2.1.0\lib\Release\x64`" folder to the "`<BASE_DIR>\lib`" folder.

Set the user environment variable "`KVS_GLEW_DIR`" to the value "`<BASE_DIR>\lib\glew-2.1.0`".

3.4.3. KVS2.9mod4meta

Set the user environment variable "`KVS_DIR`" to the value "`<BASE_DIR>\lib\kvs`". KVS will be installed in this folder.

Decompress the `KVS2.9mod4meta.zip` into the "`KVS2.9mod4meta`" folder and move it to the "`<BASE_DIR>`" folder. Launch the x64 Native Tools command prompt for Visual Studio (hereinafter referred to as "Tools command prompt"). The version of Visual Studio is replaced as needed.

On the Tools command prompt, navigate to the "`<BASE_DIR>\KVS2.9mod4meta`" folder and execute the following command.

```
nmake
```

```
nmake install
```

Add the value "%KVS_DIR%\bin " to the user environment variable "Path".

3.5. CGFormatExt4KVS and Dependency

3.5.1. Assimp

Decompress the "assimp-5.0.0.zip" as "assimp-5.0.0" folder and move it to the "<BASE_DIR>\lib" folder.

At the Tools command prompt, move to the "<BASE_DIR>\lib\assimp-5.0.0" folder and execute the following command. The version of Visual Studio is replaced as needed.

```
> cd assimp-5.0.0
> SET CMAKE_BIN=<BASE_DIR>\tools\cmake-3.16.3-win64-x64\bin\cmake.exe
> 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 Visual Studio\2017\Community
> %CMAKE_BIN% CMakeLists.txt -G "%GENERATOR%" -A x64 -D CMAKE_GENERATOR_INSTANCE="%CMAKE_GENERATOR_INSTANCE%" -D CMAKE_GENERATOR="%CMAKE_GENERATOR%" -S %SOURCE_DIR% -B %BINARIES_DIR%
> %CMAKE_BIN% --build %BINARIES_DIR% --config debug
> %CMAKE_BIN% --build %BINARIES_DIR% --config release
```

Copy

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

to

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

after building, the following library files are generated in "assimp-5.0.0\BINARIES\x64\code\Release / Debug".

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

Set following two environmental variables.

Variables	Values
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

Install the FBX SDK in <FBX_SDK_DIR> using Autodesk_FBX_Review_Win_64bit.exe. Then set the following two user environment variables. The version of Visual Studio is replaced as needed.

Variables	Values
FBX_SDK_INC_DIR	<FBX_SDK_DIR>%include
FBX_SDK_LIB_DIR	<FBX_SDK_DIR>%lib%vs2017%x64%release

3.5.3. CGFormatExt4KVS

Rename "kvsmake_libs.vc.conf_template" in the "CGFormatExt4KVS" folder to "kvsmake_libs.vc.conf" and set the library path to "FBX_SDK_DIR" and "ASSIMP_DIR" respectively. This path may contain spaces, so it should be enclosed in " ".

If Assimp and FBX are unnecessary, edit variables defined in kvsmake_libs.vc.conf as following.

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

At the Tools command prompt, go to the "<BASE_DIR>%lib%CGFormatExt4KVS%Lib" folder and execute the following command.

```
kvsmake lib
```

After the command execution, "LibCGFormatExt4KVS.lib" will be created in the "<BASE_DIR>%Lib%CGFormatExt4KVS%Lib" folder.

Set the environment variable "CGFORMAT_EXT4KVS_SHADER_DIR" to the value

"<BASE_DIR>%lib%CGFormatExt4KVS%Lib

3.6. ML-PBVR and Dependency

3.6.1. Oculus SDK

Create the folder "ovr_sdk_win_1.30.0_public" in the "<BASE_DIR>%lib" folder. unzip the ovr_sdk_win_1.30.0_public.zip and copy all files and folders in it into the "<BASE_DIR>%lib% Copy all the files and folders in the "<BASE_DIR>%lib%1.30.0_public%" folder.

The version of Visual Studio is replaced as needed.

Then set the following two user environment variables

変数名	変数值
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

Decompress imgui-1.79.zip into the "imgui-1.79" folder and move it to the "<BASE_DIR>%lib" folder. imgui-1.79_cmakefiles.zip into the "imgui-1.79" folder and move it to the "<BASE_DIR>%lib Move it to the "<BASE_DIR>%imgui-1.79" folder.

At the Tools command prompt, move to the "<BASE_DIR>%lib%imgui-1.79" folder and execute the following command. The version of Visual Studio is replaced as needed.

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

After the command execution, "libimgui.lib" and "libimgui_impl_opengl3.lib" is created in the "<BASE_DIR>%lib%imgui-1.79" folder.

3.6.3. CS-IS-PBVR

Please clone the source code of CS-IS-PBVR from github (<https://github.com/CCSEPBVR/CS-IS-PBVR>) to <BASE_DIR>. In order to build source code with VR mode, edit config file in %CS-IS-PBVR%QtClient%qtpbvr.conf as following.

3.6.3.1. Project settint

Open the project file "<BASE_DIR>%CS-IS-PBVR%QtClient%QtClient.pro" in

QtCreator. On the project settings screen, click "Desktop Qt 5.9.9 MSVC2017 64bit" > "Build" under "Build & Run" on the left sidebar to display "Build Settings" (Figure 3.6.3-1).

The version of Visual Studio is replaced as needed.

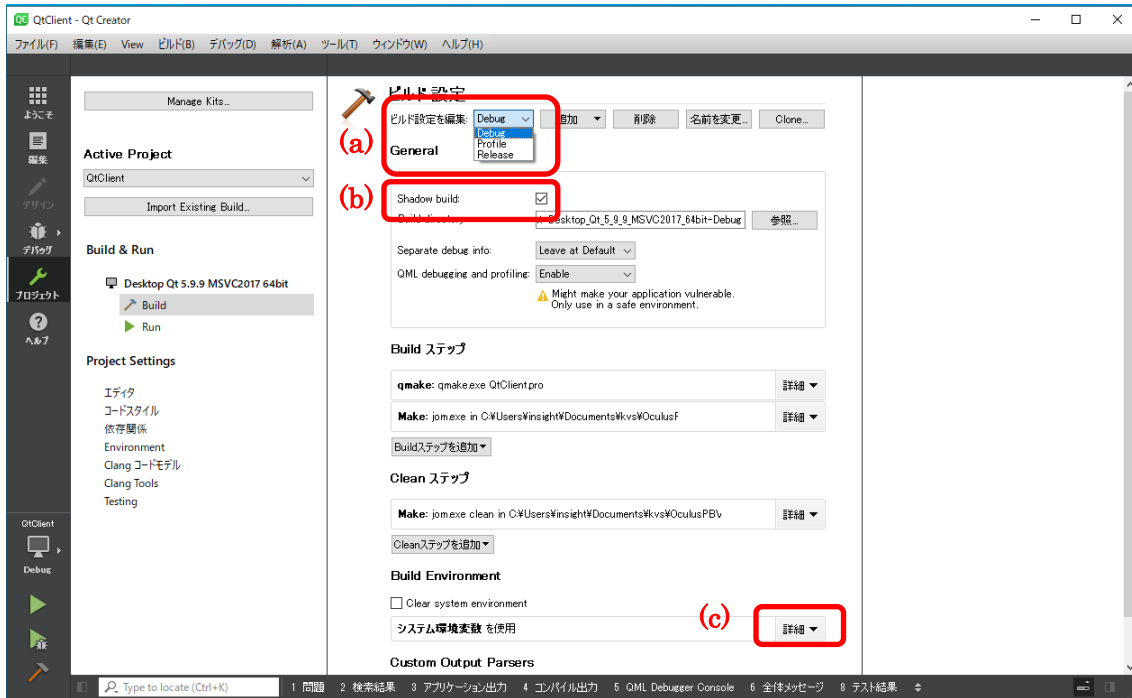


Figure 3.6.3-1 QtCreator build setting

(1) Build Setting

Select "Release" in the "Edit build configuration" (Figure 3.6.3-1 (a)).

(2) General > Shadow Build

Check "General > Shadow Build" (Figure 3.6.3-1 (b)).

(3) Build Environment

Click on "Details" (Figure 3.6.3-1 (c)) under "Build Environment" and add the following environment variables.

Variable	Value
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. Configuration

The configuration of ML-PBVR can be set by the value of the DEFINES variable in the SETTINGS.pri file.

(1) Mirroring

ML-PBVR mirrors the HMD image to the GUI on the display; the default value of DEFINES is (a).

- (a) `DEFINES += MIRROR_IMAGE_BOTH_DISTORTED`
For binocular images with distortion
- (b) `DEFINES += MIRROR_IMAGE_BOTH`
For binocular images without distortion
- (c) `DEFINES += MIRROR_IMAGE_LEFT_ONLY`
For left eye image
- (d) `DEFINES += MIRROR_IMAGE_RIGHT_ONLY`
For right eye image

(2) Coordinate Transformation

The Touch controller grabs the visualization object and transforms the coordinates of the object with its movement. The grabbing motion is tied to the Touch controller's HandTrigger or IndexTrigger. The default value of DEFINES is (a).

- (a) `DEFINES += GRAB_USING_HAND_TRIGGER`
Define a grabbing action on the middle finger button (HandTrigger).
- (b) `DEFINES += GRAB_USING_INDEX_TRIGGER`
Define the action of grabbing on the index finger button (IndexTrigger).

(3) Debug Log output

Output processing logs of Scene, Screen, and TouchController classes to standard output.

- (a) Scene class: `“DEFINES += DEBUG_SCENE”`
- (b) Screen class : `“DEFINES += DEBUG_SCREEN”`
- (c) TouchController class : `“DEFINES += DEBUG_TOUCH_CONTROLLER”`

3.6.3.3. Build

Menu bar > Build > Build project “QtClient”

3.7. Server Program

CS-PBVR consists of the filter program that divides large data into sub-regions for efficient parallel processing, the Particle Sampler that converts volume data into particles for visualization on the server, and the Particle Renderer that renders the volume on the user PC. ML-PBVR is a VR extension of the Particle Renderer, and the filter program and Particle Sampler are common to CS-PBVR. For details, please refer to CS-PBVR manual (<https://ccse.jaea.go.jp/software/PBVR/>) for more details.

4. Build procedure

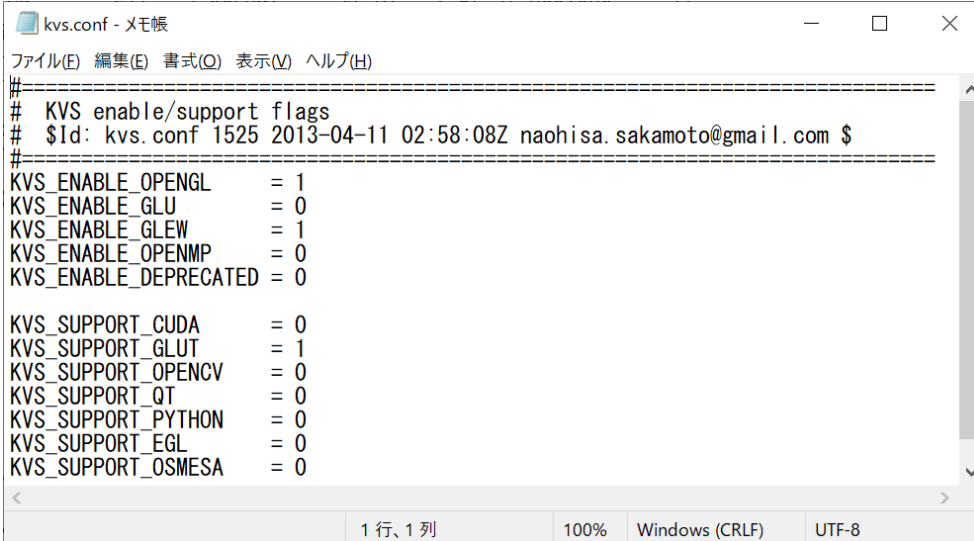
The build procedure for this program is summarized in this section.

Launch x64 Native Tools Command Prompt for VS 2019.

Clone the source code of the client program and KVS library from CCSE's git.

```
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
```

Edit `D:\WorkDIR\KVS\KVS2.9mod4meta\kvs.conf` as following figure.

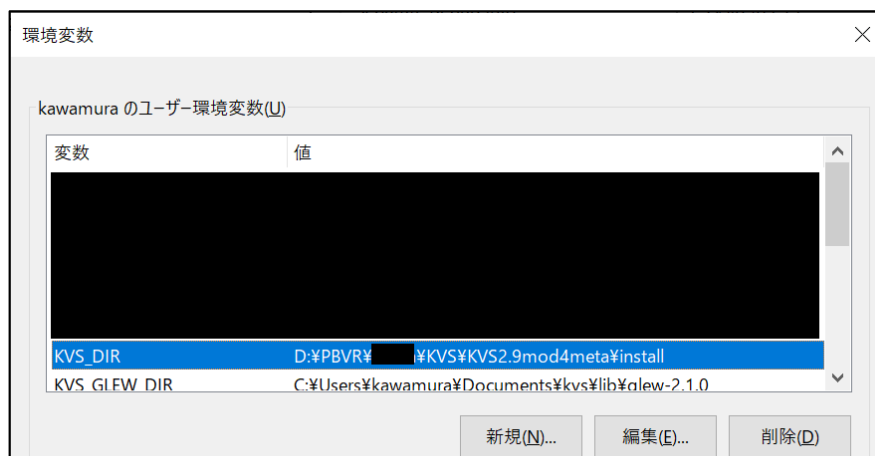


```
kvs.conf - メモ帳
ファイル(E) 編集(E) 書式(O) 表示(V) ヘルプ(H)
#=====#
# KVS enable/support flags
# $Id: kvs.conf 1525 2013-04-11 02:58:08Z naohisa.sakamoto@gmail.com $
#=====#
KVS_ENABLE_OPENGL      = 1
KVS_ENABLE_GLU         = 0
KVS_ENABLE_GLEW        = 1
KVS_ENABLE_OPENMP      = 0
KVS_ENABLE_DEPRECATED  = 0

KVS_SUPPORT_CUDA       = 0
KVS_SUPPORT_GLUT       = 1
KVS_SUPPORT_OPENCV     = 0
KVS_SUPPORT_QT         = 0
KVS_SUPPORT_PYTHON     = 0
KVS_SUPPORT_EGL        = 0
KVS_SUPPORT_OSMESA     = 0

1行、1列 100% Windows (CRLF) UTF-8
```

Set the environment variable `KVS_DIR` as following.



Build and install KVS library.

```
nmake  
nmake install
```

Install Assimp library.

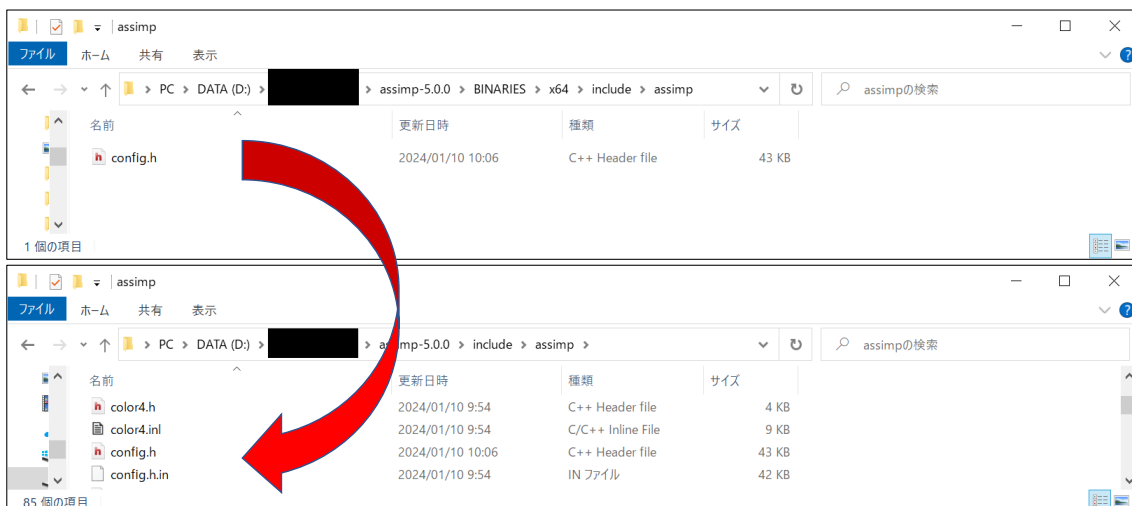
```
> SET SOURCE_DIR=.  
  
> SET GENERATOR=Visual Studio 16 2019  
  
> SET BINARIES_DIR="./BINARIES/x64 "  
  
> SET CMAKE_GENERATOR_INSTANCE=C:\Program Files (x86)\Microsoft Visual Studio\2019\Community  
  
> cmake CMakeLists.txt -G "%GENERATOR%" -A x64 -D CMAKE_GENERATOR_INSTANCE="%CMAKE_GENERATOR_INSTANCE%" -D CMAKE_GENERATOR="%CMAKE_GENERATOR%" -S %SOURCE_DIR% -B %BINARIES_DIR%  
  
> cmake --build %BINARIES_DIR% --config debug  
  
> cmake --build %BINARIES_DIR% --config release
```

Move Assimp library from

D:\WorkDIR\assimp-5.0\BINARIES\x64\include\assimp\config.h

to

D:\WorkDIR\assimp-5.0\include\assimp



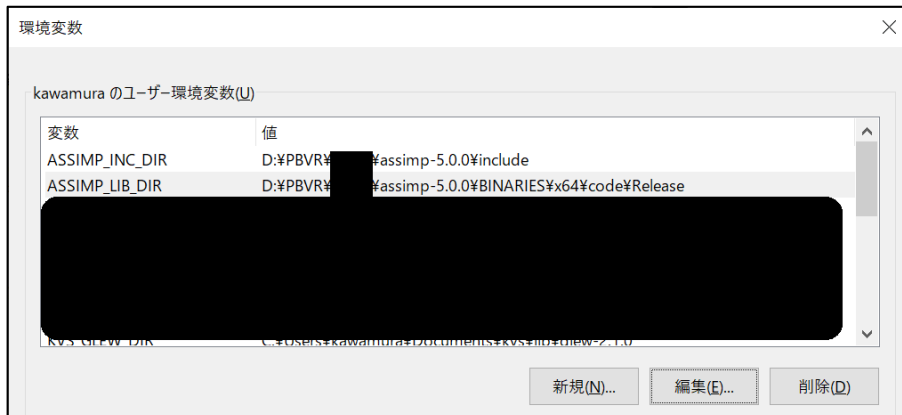
Confirm the library files are generated in

D:¥WorkDIR¥assimp-5.0.0¥BINARIES¥x64¥code¥Release and

D:¥WorkDIR¥assimp-5.0.0¥BINARIES¥x64¥code¥Debug

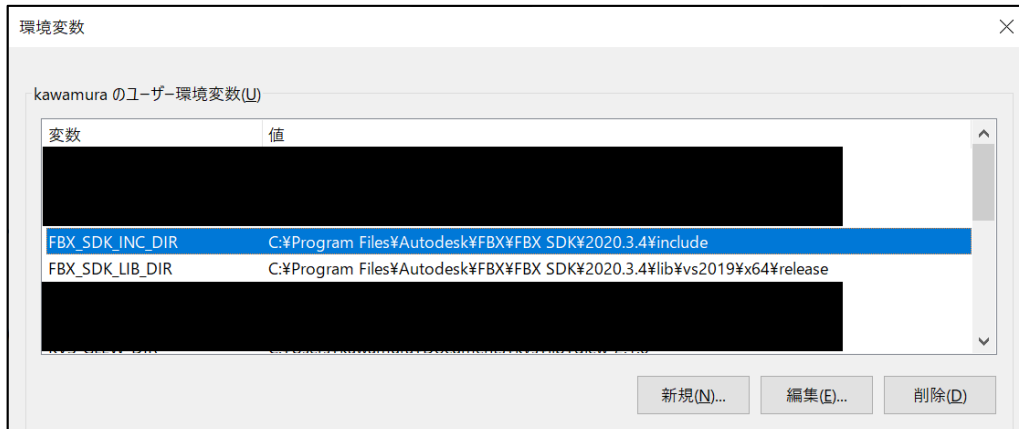


Set the environmental variables ASSIMP_INC_DIR and ASSIMP_LIB_DIR.



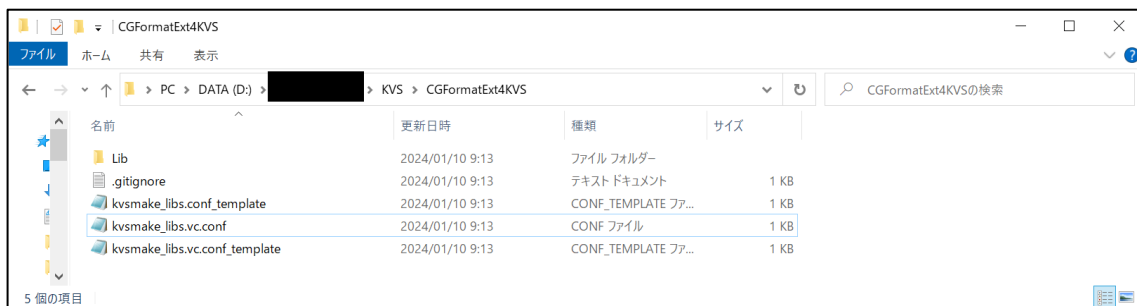
Download FBX SDK and install according to the instruction of the installer. In this manual, the path of the install directory is default.

Set the environmental variables FBX_SDK_INC_DIR and FBX_SDK_LIB_DIR.

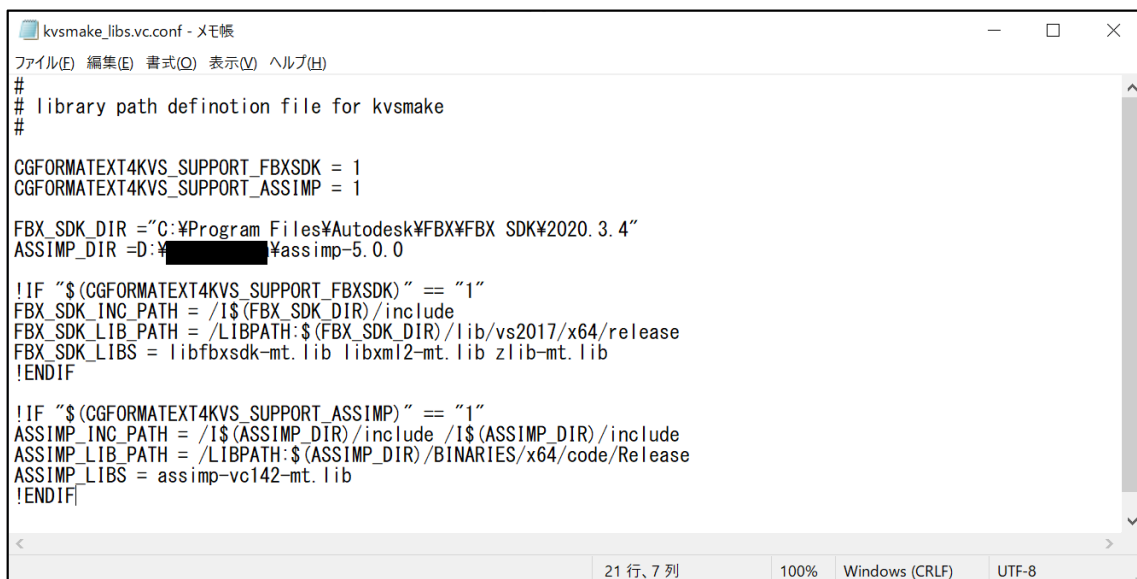


Rename

D:\WorkDIR\KVS\CGFormatExt4KVS\kvsmake_libs.vc.conf_template
into kvsmake_libs.vc.conf.



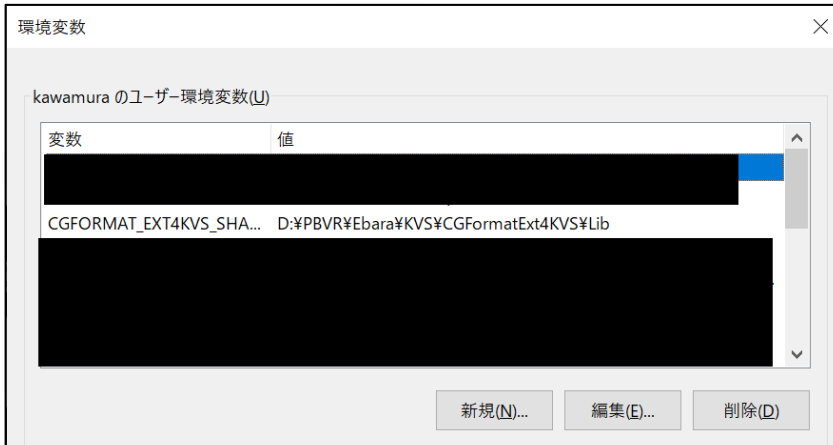
Edit kvsmake_libs.vc.conf and add settings of ASSIMP and CGFormatExt4KVS.



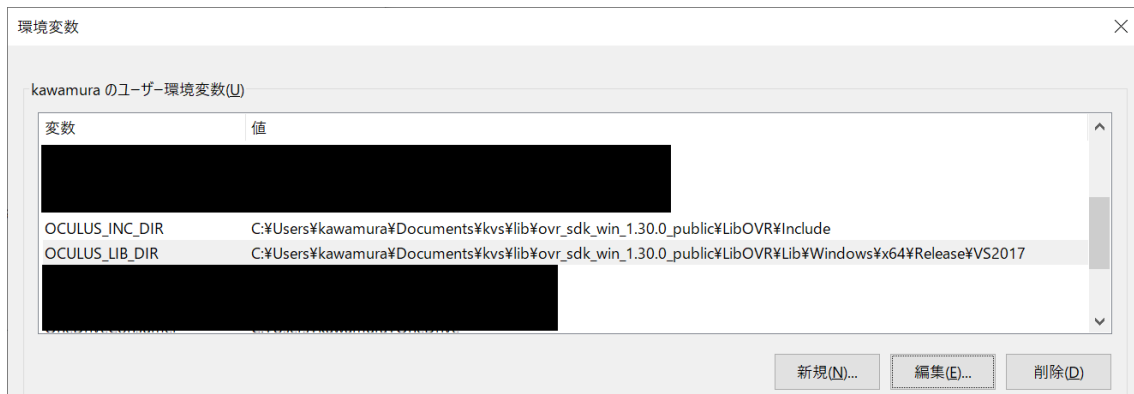
Build CGFormatExt4KVS.

```
kvsmake lib
```

Set the environmental variable CGFORMAT_EXT4KVS_SHADER_DIR.



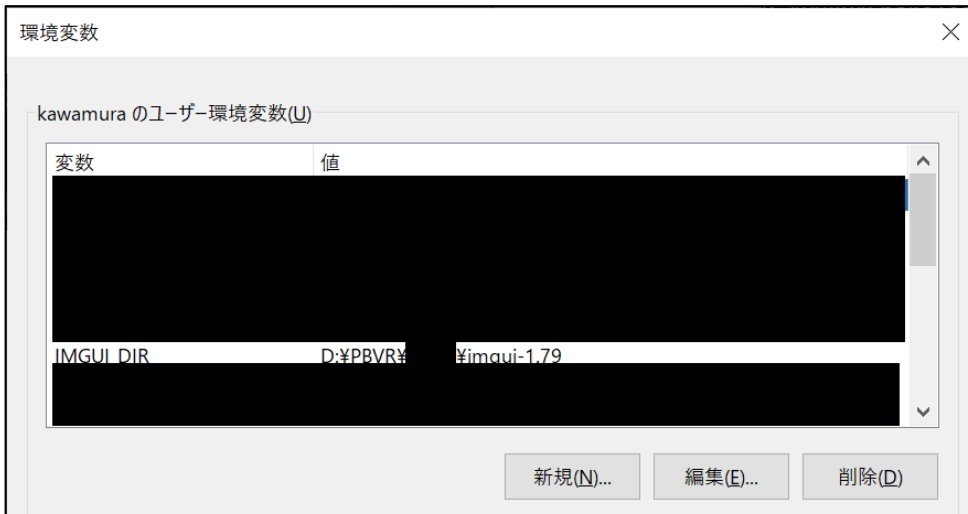
Download the Oculus SDK and set the environment variables OCULUS_INC_DIR and OCULUS_LIB_DIR. Be sure to use v1.30.0. If you build the client using v32.0.0, you will get an error "Cannot initialize LibOVR".



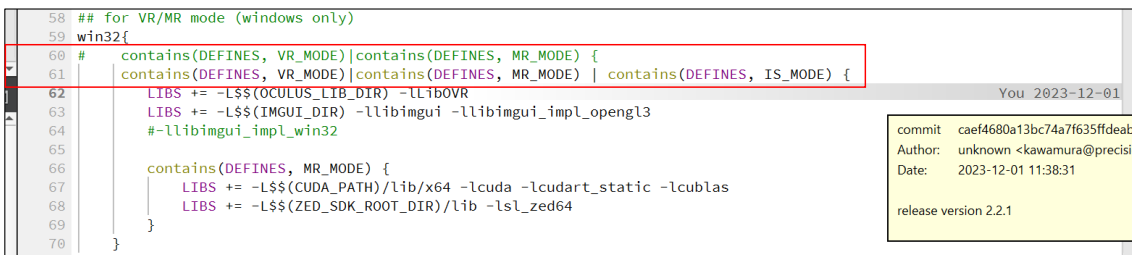
Download ImGui and install it.

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

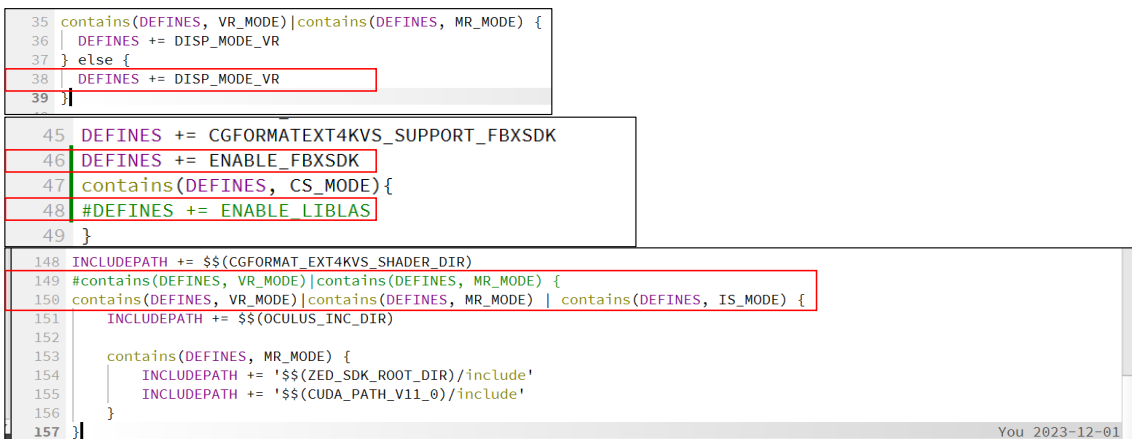
Set the environmental variable `IMGUI_DIR`.



Edit lines 60 and 61 of `App.pro` as following figure in `D:\WorkDIR\CS-IS-PBVR\QtClient\QtClient.pro` using QtCreator.



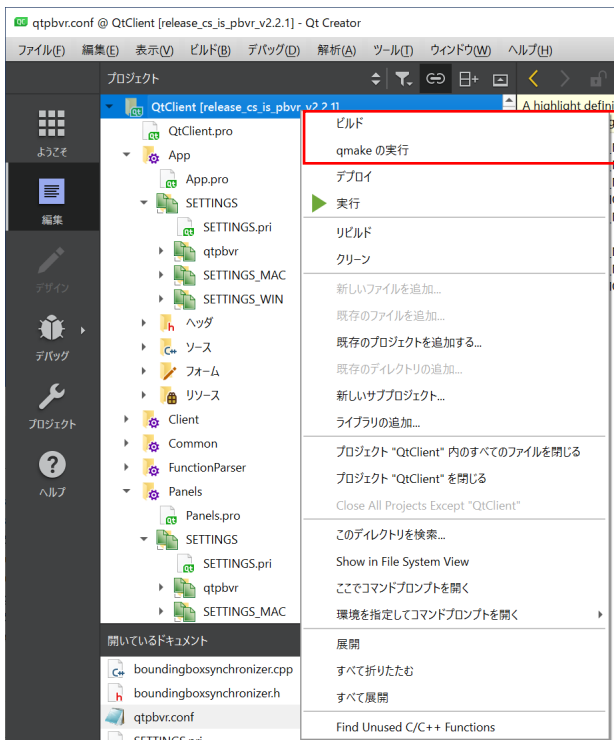
Edit lines 38,46,48,149,150 of `SETTING.pri` as following figure in `D:\WorkDIR\CS-IS-PBVR\QtClient\QtClient.pro` using QtCreator.



Edit qtpbvr.conf as following figure.

```
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
1 #PBVR_MODE = CS (ClientServer), or IS (Insitu) or VR (Virtual Reality) or MR (Mixed Reality) - Needed on all platforms
2 #PBVR_MODE = CS
3 PBVR_MODE = IS
4 #PBVR_MODE = VR
5 #PBVR_MODE = MR
6
7 #REND_MODE = CPU, or GPU - Needed on all platforms
8 #REND_MODE = CPU
9 REND_MODE = GPU
10
11
```

Qmake and build on QtCreator.



5. Launching ML-PBVR

ML-PBVR uses a common server with IS-PBVR; ML-PBVR users are classified as hosts and guests. The host is the subject of the analysis and is responsible for specifying visualization parameters, while the guest receives the visualization results.

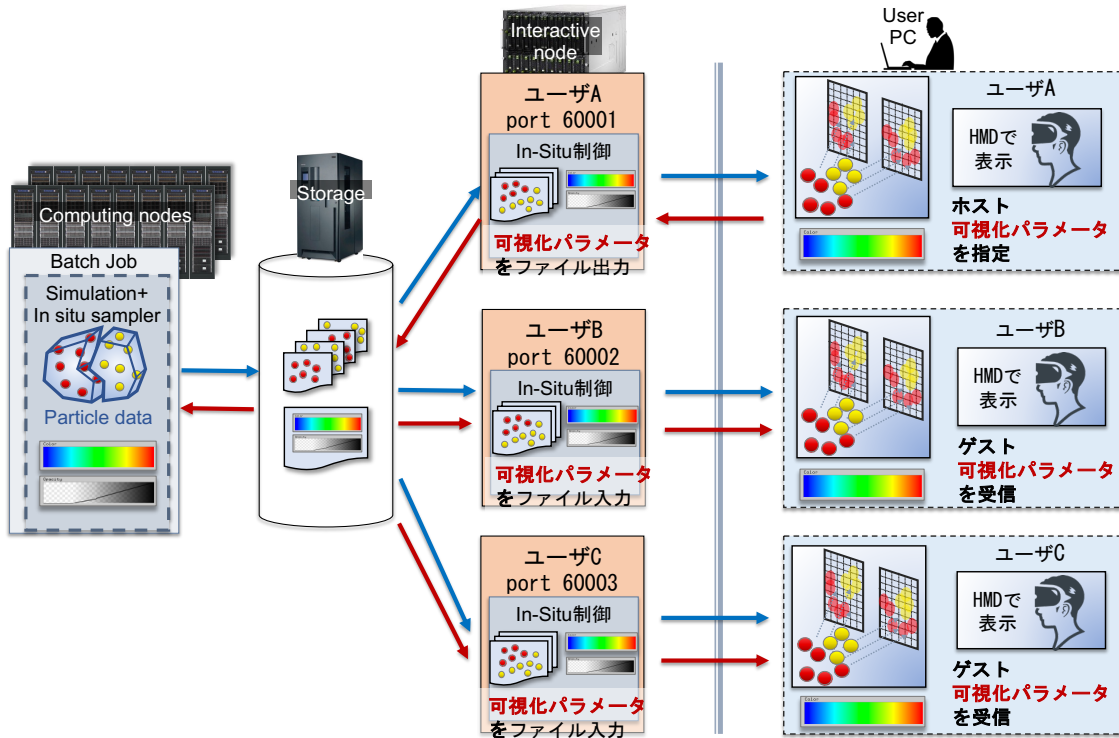


Figure 3.6.3-1 Components of ML-PBVR

This section explains how to launch ML-PBVR. Please refer IS-PBVR manual to launch the server. ML-PBVR is started from the project file." Command line arguments can be entered from the "Run-Time Settings" screen opened from "Build & Run" > "Desktop Qt 5.9.9 MSVC2017 64bit" > "Run". The version of Visual Studio is replaced as needed. ML-PBVR is run from the button under the left sidebar or from the menu bar. The following options are available as command line arguments

- -vin (Specify the visualization data file name)
- -tf (Specify the transfer function file name)
- -lefty (Left-handed configuration)
- -cgmodel <FBX file or 3ds file> (Display FBX or 3ds files)
- Other options available, same as CS-PBVR.

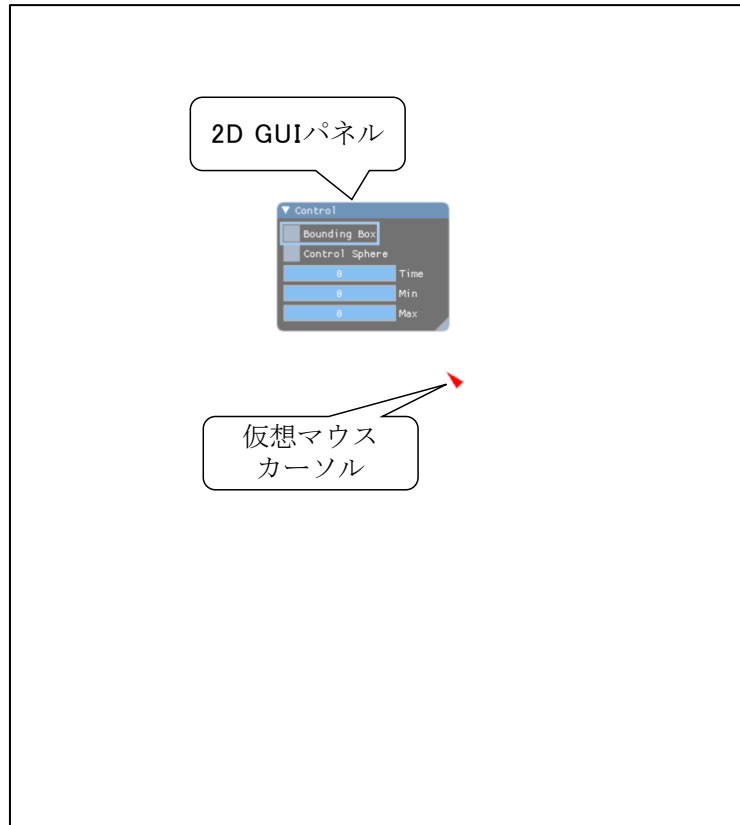


Figure 3.6.3-2 Control panel and mouse cursor immediately after startup

Immediately after execution, a control panel displayed in layover and a virtual mouse cursor for manipulating it are drawn in the 3D space (Figure 3.6.3-2).

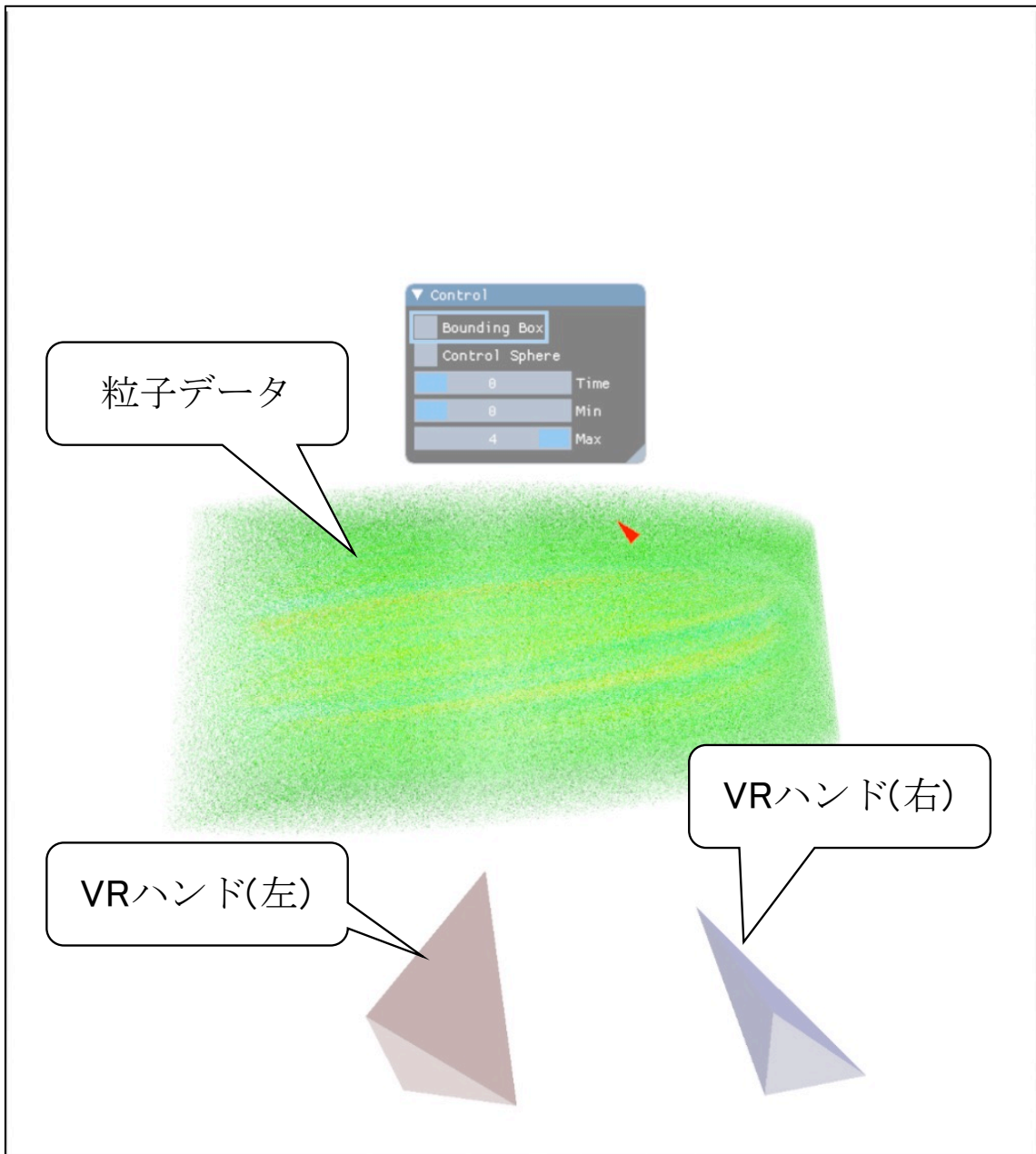


Figure 3.6.3-3 Visualization data and VR hand

The renderer receives particle data from the server and draws visualization data and VR hands (Figure 3.6.3-3).

6. Operation in VR space

6.1. Control Panel

The control panel can be operated with the red triangular mouse cursor. In right-handed mode, the mouse cursor is moved with the left stick of the Touch Controller and clicked with the A button (Figure 3.6.3-1). In left-handed mode, the mouse cursor is moved with the right stick and clicked with the X button (Figure 3.6.3-2).



Figure 3.6.3-1 Right-handed mode (without "-lefty" option)



Figure 3.6.3-2 Left-handed mode (with "-lefty" option)

The control panel (Figure 3.6.3-3) provides the following functions

- (1) Toggling the bounding box display On/Off (Figure 5.1 4)
- (2) Toggle control sphere display On/Off (Figure 5.1 5)
- (3) Time step setting
 - (a) Designation of display step
 - (b) Designation of minimum step
 - (c) Designation of maximum step

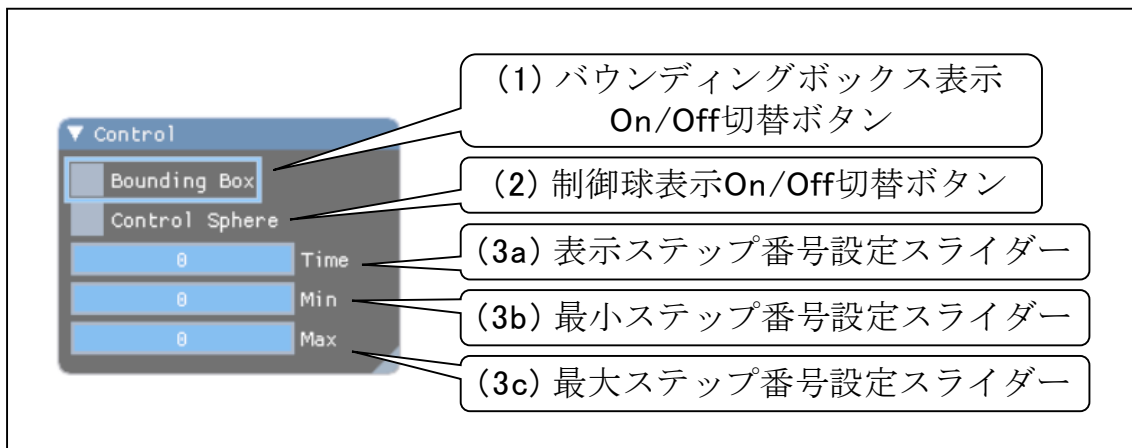


Figure 3.6.3-3 GUI panel operation items

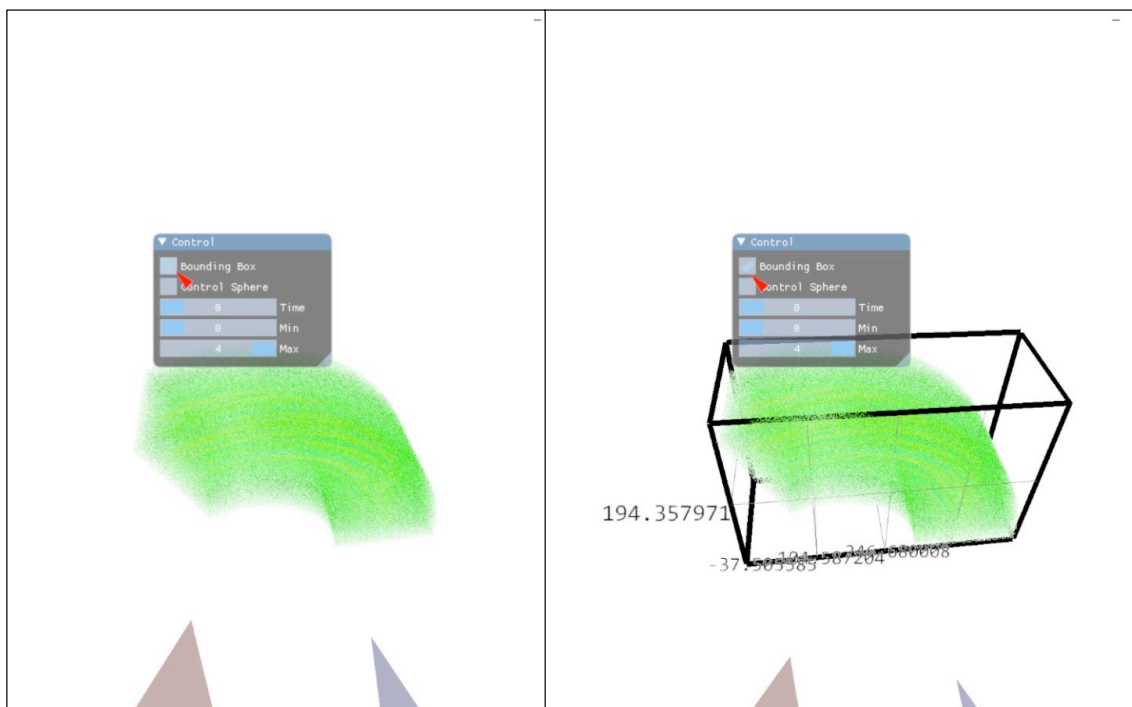


Figure 3.6.3-4 Bounding box display Off (left) and On (right)

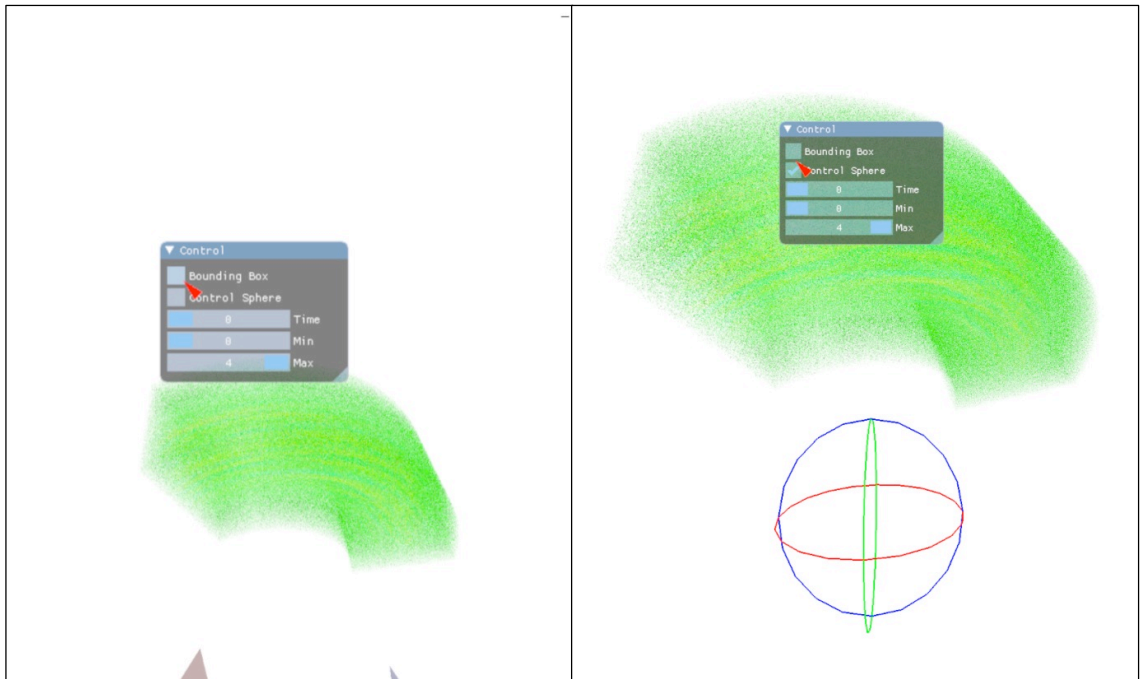


Figure 3.6.3-5 Control ball display Off (left) and display On (right)

For time steps, the top slider is the display step setting, the next is the lower limit of steps that can be displayed, and the bottom is the upper limit of steps that can be displayed. The time steps on the control panel and the GUI on the display are interlocked (Figure 3.6.3-6).

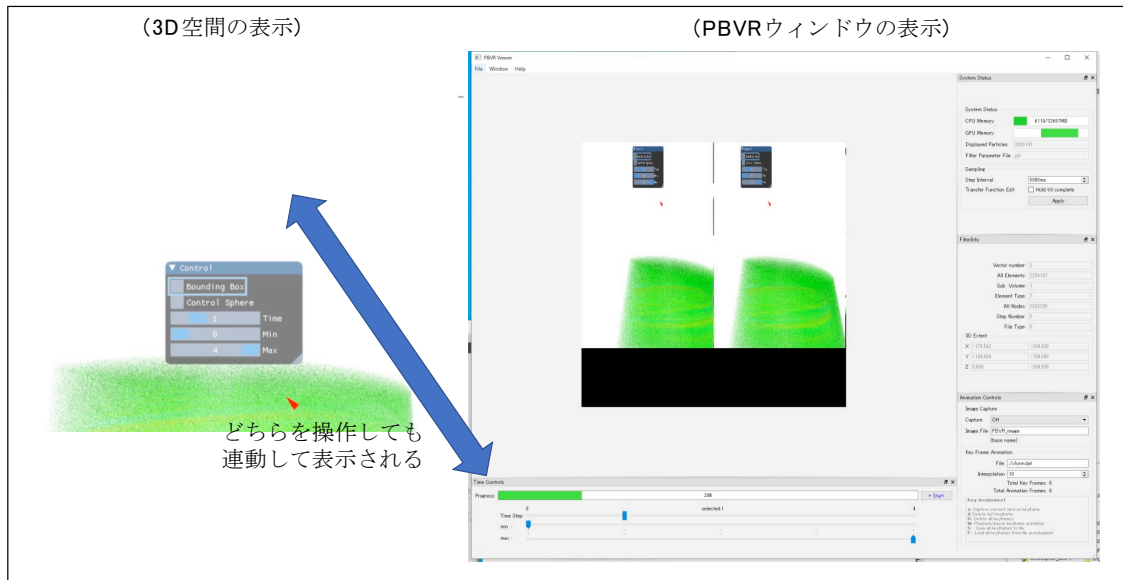


Figure 3.6.3-6 VR space and GUI on display

6.2. Coordinate Transformation

The object is grabbed by the trigger button on the Touch controller and the coordinates are transformed by the gesture. The trigger button is initially HandTrigger (Figure 3.6.3-1).



Figure 3.6.3-1 Trigger button position

Gestures that move both hands in the same direction cause the object to move in parallel (Figure 3.6.3-2).



Figure 3.6.3-2 Parallel translation

Gestures of releasing or moving both hands closer together cause the object to expand or contract (Figure 3.6.3-3).

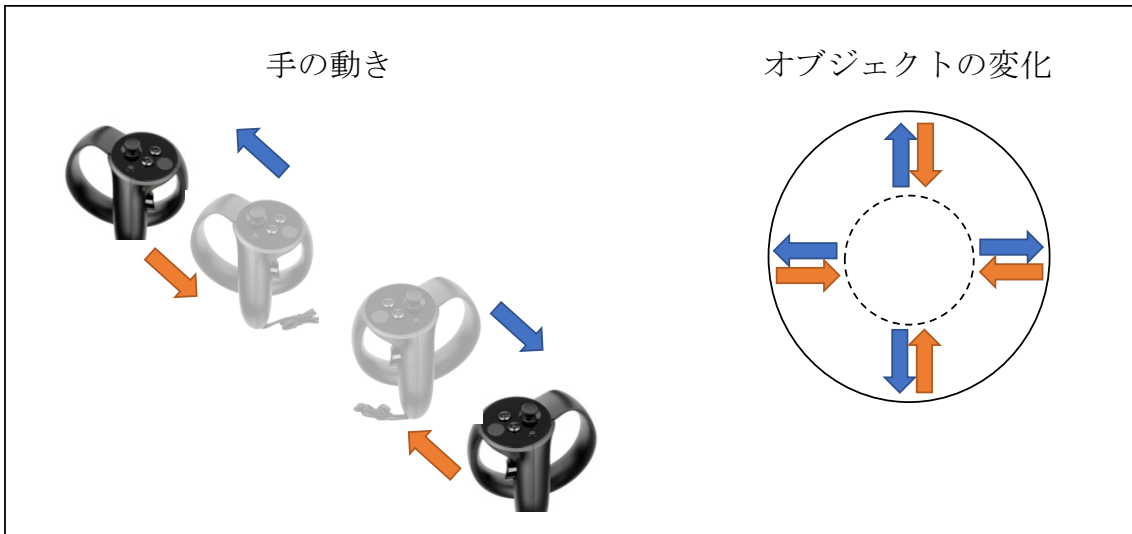


Figure 3.6.3-3 Scaling

The gesture of turning both hands rotates the object (Figure 3.6.3-4).

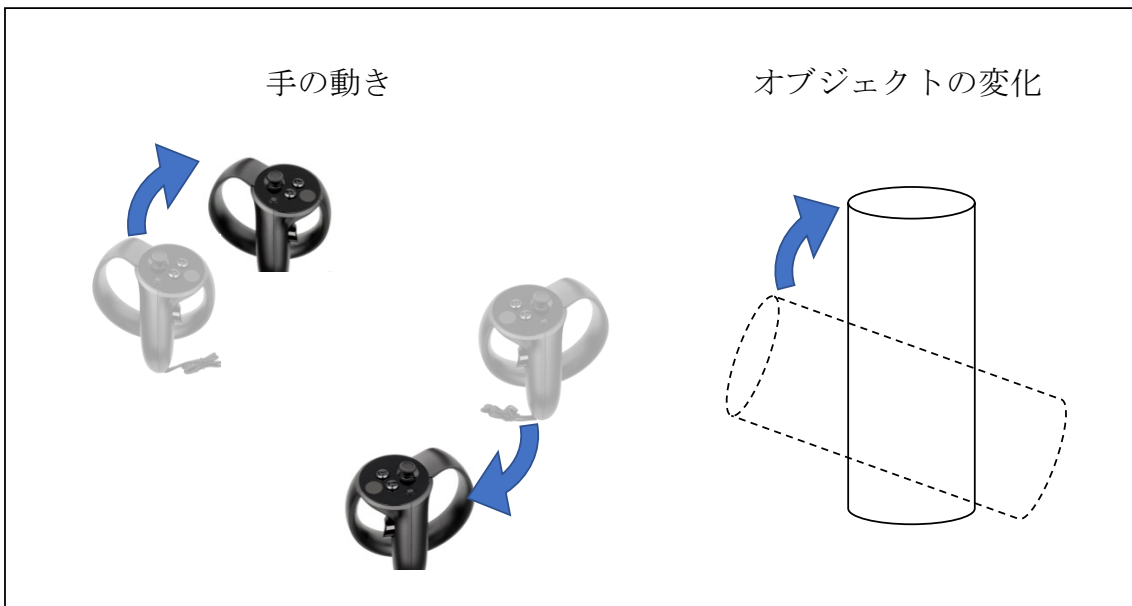


Figure 3.6.3-4 Rotation

7. OculusPBVR_JAEA の終了

The client exits by pressing the X button in the upper left corner of the main window on the display. The server terminates by typing Ctrl+C on the terminal.