

MIREVI MotionHub Developer Documentation

Hochschule Düsseldorf
University of Applied Sciences



Fachbereich Medien
Faculty of Media



SPONSORED BY THE



Federal Ministry
of Education
and Research

MIREVI MotionHub (MMH) is a middleware for merging body tracking data from different systems into one coordinate space in real-time in order to combine and use their individual benefits.

MMH offers support for several body tracking systems and encompasses a game engine plug-in that connects the MMH with Unity by means of a standardized protocol. The plug-in allows for the usage of a single type of skeleton for any body tracking system and, therefore, facilitates the switch between different body tracking systems during app development significantly.

MotionHub is developed at the research lab **MIREVI** from the [University of Applied Sciences Düsseldorf](#) within the scope of the project [HIVE](#).

Acknowledgements

The project [HIVE](#) is sponsored by the [German Federal Ministry of Education and Research](#) (BMBF) under the project number **16SV8182**.

Content

1. [Setup and Building](#)
2. [Class Collaboration Diagram](#)
3. [Tracking Loop](#)
4. [Implement a new Tracker](#)
5. [Skeleton OSC Data Structure](#)
6. [Azure Kinect Offset Values](#)

1. Setup and Building

The [CMake](#) system is used to generate project files and for downloading all required dependencies. Please use the `CMakeLists.txt` file for generating.

- MMH is developed with Microsoft **Visual Studio 2017**. (*[CMake](#) has only been tested with this IDE version.*)

1. Download or clone MMH
2. Download Azure [Kinect Sensor](#) and [Body Tracking](#) SDK
3. In CMake, set source to the MMH path
4. Set binaries to (MMH path)/build
5. Click "configure" (click Yes, select your installed VS version and Finish). This will take a few minutes (and is expected to FAIL) so go on with 5 and 6.
6. Download [Azure Kinect Sensor SDK](#)

7. Download [Azure Kinect Body Tracking SDK](#)
8. Install both and remember the install paths!
9. Meanwhile, the configuration process in CMake should have failed, because it couldn't find Azure Kinect dependencies.
10. In CMake, set the flags "advanced" and "grouped", you should see all dependencies listed, including K4A (Sensor SDK) and K4ABT (Body Tracking SDK)
11. Under K4A set K4A_INCLUDE_DIR to the include folders path (something like this in your freshly installed directory: /Azure Kinect SDK v1.4.0/sdk/include)
12. Repeat for K4A_LIBRARIES_DIR (/Azure Kinect SDK v1.4.0/sdk/windows-desktop/amd64/release/lib).
13. Repeat both for K4ABT (/AzureKinectBodyTrackingSDKv1.0.1/sdk/include) and (/AzureKinectBodyTrackingSDKv1.0.1/sdk/windows-desktop/amd64/release/lib)
14. Again, click "Configure" (This should be successful), then click "Generate".
15. You can now click "Open Project" or open the Solution with VS (/MMH/build/MireviMotionHub.sln)
16. In the VS Solution Explorer, right click on the project "MireviMotionHub"->"Set as StartUp Project" and "MireviMotionHub"->"build"
17. After the code is compiled, you can push F5 (Local Windows Debugger) in VS or execute /MMH/build/bin/MireviMotionHub.exe to start the MotionHub

Please Note that we use Qt Framework for the UI. To build the project you need the [Qt Visual Studio Tools](#) and for editing Qt .ui files you need [Qt Designer](#).

2. Class Collaboration Diagram

For a software architecture overview please see the `mmhClassCollaborationDiagram` file in the `doc` folder.

3. Tracking Loop

The main loop method named `update()` is located in the `MotionHub` class. Following order is executed by the main loop for every cycle.

1. process the user input
2. update the console
3. get skeleton pool from each Tracker
4. update the render window
5. send skeleton pool via OSC
6. update the main window
7. start a new tracking cycle

All tracker run on there own thread. After each tracking cycle the `MotionHub` class gets the skeleton pool and resets each trackers flag to start a new tracking cycle.

4. Implement a new Tracker

Follow the listed steps to implement a new tracker in the MMH.

1. create class files in the explorer under `src/TrackerManagement` (please follow the name convention "trackerAcronym + Tracker" e.g AKTracker for Azure Kinect Tracker)
2. run CMake to add them to the Visual Studio project
3. inherence your new class from the `Tracker` base class
4. implement the pure virtual methods `start()` , `stop()` and `destroy()` (please see other specific tracker classes for implementation reference)
5. include the new tracker in `TrackerManager.h`
6. add the new tracker to the UI and connect the signals for user interaction

6.1 open `CreateTrackerWindow.ui` in `RenderManagement` with [Qt Designer](#)

6.2 open `dropdown_tracker`

6.3 add a new item with the tracker name at the end of the list

6.4 note the index of the item in the list (starting from 0 at the top)

6.5 save and close the `CreateTrackerWindow.ui` file

6.6 add the tracker name with the correct index position in the `TrackerType` enum in `TrackerManager.h`

6.7 add the tracker with the correct index position in the `createTracker()` method in `TrackerManager.cpp` and implement the logic (please see other tracker for implementation reference)

5. Skeleton OSC Data Structure

Follow data is send by the `NetworkManager` with the OSC protocol to localhost.

The coordinate system is right handed and all position values are in meters.

Index	Name	DataType	DataStructue
0	trackerID + skeletonID	int	e.g. 1001 -> 1 is trackerID 001 is skeletonID
1 - 8	HIPS	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
9 - 16	SPINE	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
17 - 24	CHEST	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
25 - 32	NECK	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
33 - 40	SHOULDER_L	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
41 - 48	ARM_L	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
49 - 56	FOREARM_L	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]

Index	Name	DataType	DataStructure
57 - 64	HAND_L	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
65 - 72	SHOULDER_R	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
73 - 80	ARM_R	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
81 - 88	FOREARM_R	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
89 - 96	HAND_R	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
97 - 104	UPLEG_L	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
105 - 112	LEG_L	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
113 - 120	FOOT_L	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
121 - 128	TOE_L	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
129 - 136	UPLEG_R	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
137 - 144	LEG_R	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
145 - 152	FOOT_R	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
153 - 160	TOE_R	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
261 - 268	HEAD	7 float + 1 int	positionX, positionY, positionZ, quaternionX, quaternionY, quaternionZ, quaternionW, confidence enum (0 - 3) [NONE, LOW, MEDIUM, HIGH]
269	skeleton posture	int	posture enum (0 - 5) [UNKNOWN, STAND, SIT, CROUCH, JUMP, LIE]

6. Azure Kinect Offset Values

Listed offset quaternions for each joint can be used to transform them into a right handed coordinate system.

Index	Name	Quaternion Offset Value (x,y,z,w)
0	HIPS	(0.500000, -0.500000, 0.500000, 0.500000)
1	SPINE	(0.500000, -0.500000, 0.500000, 0.500000)
2	CHEST	(0.500000, -0.500000, 0.500000, 0.500000)

Index	Name	Quaternion Offset Value (x,y,z,w)
3	NECK	(0.500000, -0.500000, 0.500000, 0.500000)
4	SHOULDER_L	(0.707107, 0.000000, -0.000000, 0.707107)
5	ARM_L	(0.707107, 0.000000, -0.000000, 0.707107)
6	FOREARM_L	(0.707107, 0.000000, -0.000000, 0.707107)
7	HAND_L	(1.000000, -0.000000, 0.000000, 0.000000)
8	SHOULDER_R	(0.707107, 0.000000, -0.000000, -0.707107)
9	ARM_R	(0.707107, 0.000000, -0.000000, -0.707107)
10	FOREARM_R	(0.707107, 0.000000, -0.000000, -0.707107)
11	HAND_R	(0.000000, 0.000000, -0.000000, 1.000000)
12	UPLEG_L	(0.500000, -0.500000, 0.500000, 0.500000)
13	LEG_L	(0.500000, -0.500000, 0.500000, 0.500000)
14	FOOT_L	(0.500000, -0.500000, 0.500000, 0.500000)
15	TOE_L	(0.000000, -0.707107, -0.000000, 0.707107)
16	UPLEG_R	(-0.500000, -0.500000, -0.500000, 0.500000)
17	LEG_R	(-0.500000, -0.500000, -0.500000, 0.500000)
18	FOOT_R	(-0.500000, -0.500000, -0.500000, 0.500000)
19	TOE_R	(0.707107, -0.000000, -0.707107, -0.000000)
20	HEAD	(0.500000, -0.500000, 0.500000, 0.500000)

Acknowledgment

- [bibigone: k4a.net](https://k4a.net) - For Azure Kinect local to global joint transformation quaternions.