Neuron Data Reader Runtime API

Documentation

By Yuanhui He

22.12.2014

NeuronDataReader b12

Noitom Technology Co., Ltd.

This document illustrates how user can use NeuronDataReader library and apply the bone data received by the library.

Contents

Neuron Data Reader Runtime API 1

Documentation 1

1. Overview 4

1.1. NeuronDataReader framework 4

1.1. Skeleton Data Format 4

1.2. Command and Parameter sync. 4

1.3. User Agreement 5

2. Reference 6

2.1. Data type definitions 6

2.1.1. Cross-platform data types 6

2.1.2. Socket connection status 6

2.1.3. Data version of stream 6

2.1.4. Header of BVH data stream 7

2.2. Callbacks and callback register 8

2.2.1. Skeleton data callback 8

2.2.2. Command data callback 8

2.2.3. Socket status callback 8

2.3. API reference 10

2.3.1. BRRegisterFrameDataCallback 10

2.3.2. BRRegisterCommandDataCallback 10

2.3.3. BRRegisterSocketStatusCallback 10

2.3.4. BRConnectTo 11

2.3.5. BRStartUDPServiceAt 11

2.3.6. BRCloseSocket 11

2.3.7. BRGetSocketStatus 11

2.3.8. BRCommandFetchAvatarDataFromServer 11

2.3.9. BRCommandFetchAvatarDataFromServer 12

2.3.10. BRRegisterAutoSyncParmeter 12

2.3.11. BRUnregisterAutoSyncParmeter 12

2.3.12. BRGetLastErrorMessage 12
3. Illustration 13
4. Known Bugs 23
Appendix A: Skeleton data sequence in array 24
Appendix B: BVH header template 25
Appendix C: Skeleton graph 34
Revision history 35

1. Overview

1.1. NeuronDataReader framework

The Axis Neuron software of Noitom can stream BVH motion data through TCP/IP or UDP protocol. The NeuronDataReader plugin(API library) can provide convenience for user to receive and use the BVH data stream or sync parameters by commands with server.

The structure of NeuronDataReader library is shown below.

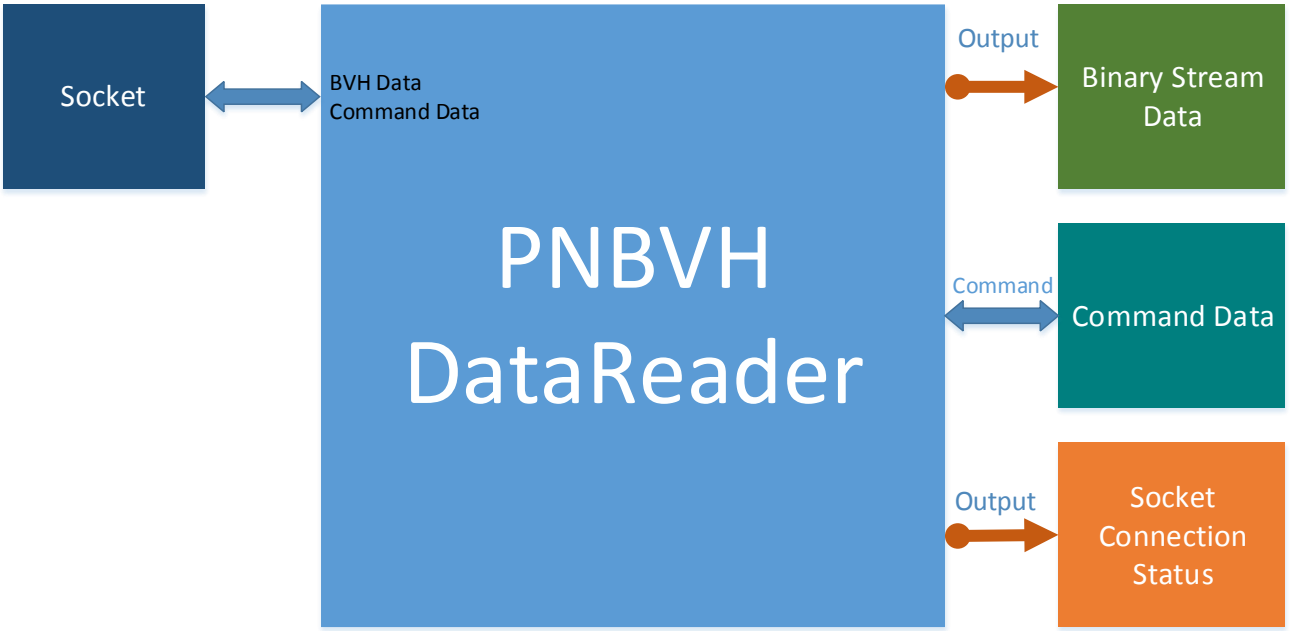


Fig. 1-1 NeuronDataReader Overview

As most other libraries, NeuronDataReader provides various C/C++ functions for users to interact with the library.

1.1. Skeleton Data Format

The action data, with BVH format, is output by callback. All information of the skeleton data, such as prefix, displacements settings, etc. are included in a `BvhDataHeader` parameter. The sequence of bone data in the float data array is shown in Appendix A. Appendix B is showing sample BVH header data, for reference in live data stream.

1.2. Command and Parameter sync.

NeuronDataReader library are mainly used to read and parser data received from server. Considering lots of parameters used at client must be sync with server, some relevant APIs are added. These APIs could register command IDs need to be automatically notified on server.

1.3. User Agreement

NeuronDataReader uses a callback method to output the data. So prior connecting to the server, a local function must be registered to receive the data. While registering the data-receiving function, the user can pass the Client Object reference into the NeuronDataReader library so that the library can output the Class Object reference along with the data stream during the callback.

The data-processing thread in the NeuronDataReader is a work thread separated from the UI. So the user-registered data-receiving function cannot access the UI elements directly. However, the data or status of the callback function can be saved into a local array or buffer, so that the UI thread can access the local-buffered data in any other place.

There are some commands in NeuronDataReader library used to sync parameters or data with server.

Since C# or Unity cannot call C++ dynamic lib API directly, NeuronDataReader uses a pure C interface.

If the NeuronDataReader lib is used in C/C++ project on Mac platform, a pre-defined symbol "__OS_XUN__" should be included in the pre-process field to import some custom-defined symbols.

2. Reference

Some data types, handles, program interfaces of NeuronDataReader lib are listed below.

2.1. Data type definitions

2.1.1. Cross-platform data types

If the NeuronDataReader lib is used in C/C++ project on Mac platform, a pre-defined symbol "__OS_XUN__" should be included in the pre-process field to import some predefined data types.

```
#ifdef __OS_XUN__
// Mac OS X, Linux or unix like OS data type definetion
typedef unsigned short      UINT16;
typedef unsigned int        UINT32;
typedef unsigned long long   UINT64;
typedef unsigned short      USHORT;
typedef unsigned char       UCHAR;
typedef unsigned char       BYTE;
typedef wchar_t             WCHAR;
#ifdef BVHDATAREADER_EXPORTS
typedef unsigned int        BOOL;
#endif
#define TRUE                1
#define FALSE               0
#define CALLBACK            // Empty
#else
#define CALLBACK            __stdcall
#endif
```

2.1.2. Socket connection status

The enumerate type below shows the socket connection status: Connected, Connecting, Disconnected.

```
// Socket status
typedef enum _SocketStatus
{
    CS_Running,             // Socket is working correctly
    CS_Starting,            // Is trying to start service
    CS_OffWork,             // Not working
}SocketStatus;
```

2.1.3. Data version of stream

For different versions of NeuronDataReader, the data structure for communication could be changed, both in

meaning and structure. Data version is used to be compatible with the data generated by old version of NeuronDataReader.

```
// Data version
typedef union BvhOutputDataVersion
{
    UINT32 _VersionMask;
    struct
    {
        BYTE BuildNumb;      // Build number
        BYTE Revision;       // Revision number
        BYTE Minor;          // Subversion number
        BYTE Major;          // Major version number
    };
} BVH_DATA_VER;
```

2.1.4. Header of BVH data stream

```
// Header format of BVH data
typedef struct _BvhDataHeader
{
    UINT16 HeaderToken1;     // Package start token: 0xDDFF
    BVH_DATA_VER DataVersion; // Version of community data format. e.g.: 1.0.0.2
    UINT32 DataCount;        // Values count, 180 for without disp data
    BOOL   WithDisp;         // With/out dispement
    BOOL   WithReference;    // With/out reference bone data at first
    UINT32 AvatarIndex;      // Avatar index
    UCHAR  AvatarName[32];   // Avatar name
    UINT32 Reserved1;        // Reserved, only enable this package has 64bytes length
    UINT32 Reserved2;        // Reserved
    UINT16 HeaderToken2;     // Package end token: 0xEEFF
} BvhDataHeader;
```

NeuronDataReader library are mainly used to read and parser data received from server. The data stream of every frame includes the BVH header and BVH motion data of float type.

BVH header is a 64 bytes header, including basic information of BVH data: whether the displacement or prefix is included.

BVH motion data is float-type array. If the data includes reference, the first 6 float number is the displacement and rotation of the reference, normally they would all be 0.

If the data includes displacement, every bone would have 6 float number: 3 displacements and 3 rotation. If the data does not include displacement, only the root node would have 6 float number (3 displacements and 3 rotation), other bones would only have 3 rotations. Please reference Appendix A for the bone sequence.

BVH data is organized as a tree structure, please reference the Appendix B for detailed BVH data structure.

2.2. Callbacks and callback register

NeuronDataReader lib outputs the skeleton data or socket status through callback functions. So related callback handles for NeuronDataReader lib should be registered firstly to receive these data.

2.2.1. Skeleton data callback

```
    typedef void (CALLBACK *FrameDataReceived)(void* customedObj, SOCKET_REF sender, BvhDataHeader* header, float* data);
```

Parameters

customedObj

User defined object.

sender

Connector reference of TCP/IP client as identity.

header

BvhDataHeader type pointer, to output the BVH data format information.

data

Float type array pointer, to output binary data.

Remarks

The related information of the data stream can be obtained from BvhDataHeader.

2.2.2. Command data callback

```
    typedef void (CALLBACK *CommandDataReceived)(void* customedObj, SOCKET_REF sender, CommandPack* pack, void* data);
```

Parameters

customedObj

User defined object.

sender

Connector reference of TCP/IP client as identity.

Pack

A command package send to or received from server.

data

Data pointer of command, related to the specified command type.

Remarks

The related information of the data stream can be obtained from command identity in pack.

2.2.3. Socket status callback

```
    typedef void (CALLBACK *SocketStatusChanged)(void* customedObj, SOCKET_REF sender, SocketStatus status, char* message);
```

Parameters

customedObj

User defined object.

sender

Connector reference of TCP/IP client as identity.

status

Indicate the status changes of current socket.

message

Status description.

Note: Since the data-processing in the NeuronDataReader is multi-threaded asynchronous, the data-receiving callback function cannot access the UI element directly. If the data need to be used in the UI thread, it is recommended to save the data from the callback function to a local array.

2.3. API reference

2.3.1. BRRegisterFrameDataCallback

Register the BVH data receiving callback handle:

```
// Register data-receiving callback handle.
BDR_API void BRRegisterFrameDataCallback(void* customedObj, FrameDataReceived handle);
```

Parameters

customedObj

User defined object.

handle

A function pointer of FrameDataReceived type.

Remarks

The handle of FrameDataReceived type points to the function address of the client.

2.3.2. BRRegisterCommandDataCallback

Command data is output in a separate channel by registering a command data callback handle.

```
// Register data-receiving callback handle.
BDR_API void BRRegisterCommandDataCallback(void* customedObj, CommandDataReceived handle);
```

Parameters

customedObj

User defined object.

handle

A function pointer of CommandDataReceived type.

Remarks

The handle of CommandDataReceived type points to the function address of the client.

2.3.3. BRRegisterSocketStatusCallback

Register socket status callback Handle:

```
// Register socket status callback
BDR_API void BRRegisterSocketStatusCallback (void* customedObj, SocketStatusChanged handle);
```

Parameters

customedObj

User defined object.

handle

A function pointer.

Remarks

The handle of SocketStatusChanged type points to the function address of the client.

2.3.4. BRConnectTo

Connect to the server with given IP address and port:

```
// Connect to server
BDR_API SOCKET_REFBRConnectTo(char* serverIP, int nPort);
```

Parameters

serverIP

Server's IP address.

nPort

Server's port.

Return Values

If connected successfully, return a handle of socket as its identity ; otherwise NULL is returned.

2.3.5. BRStartUDPServiceAt

Since Axis Neuron can output data by TCP/IP or UDP, the NeuronDataReader can read and parser the two socket data types as well. The BRStartUDPServiceAt function is used to start a service to listen and receive data sent from the server.

```
// Start a UDP service to receive data at 'nPort'
BDR_API SOCKET_REF BRStartUDPServiceAt(int nPort);
```

2.3.6. BRCloseSocket

Stop data receive service. It should be noted that it is necessary to call this function to disconnect/stop service from the server before the program exit, otherwise the program cannot exit as it is blocked by the data-receiving thread.

```
// Stop service
BDR_API void BRCloseSocket (SOCKET_REF sockRef);
```

2.3.7. BRGetSocketStatus

Check socket status. Actually the function has the same output status with the socket callback handle. If the socket status callback handle has already registered, this function is not necessary.

```
// Check connect status
BDR_API SocketStatus BRGetSocketStatus (SOCKET_REF sockRef);
```

Return Values

Return the status of refered socket.

2.3.8. BRCommandFetchAvatarDataFromServer

Send command to get avatar data from server with refered command Id. command data will be received by callback. Return FALSE if any error occurred. UDP service is not supported for now.

```
BDR_API BOOL BRCommandFetchAvatarDataFromServer(SOCKET_REF sockRef, int avatarIndex, CmdId
```

```
cmdId);
```

2.3.9. BRCommandFetchAvatarDataFromServer

Send command to get data from server with refered command Id. command data will be received by callback. Return FALSE if any error occured. UDP service is not supported for now.

```
BDR_API BOOL BRCommandFetchDataFromServer(SOCKET_REF sockRef, CmdId cmdId);
```

2.3.10. BRRegisterAutoSyncParmeter

Register parameter(s) to server for automatically notifying status changed.

```
BDR_API BOOL BRRegisterAutoSyncParmeter(SOCKET_REF sockRef, CmdId cmdId);
```

2.3.11. BRUnregisterAutoSyncParmeter

Unregister parameter(s) to server for automatically notifying status changed.

```
BDR_API BOOL BRUnregisterAutoSyncParmeter(SOCKET_REF sockRef, CmdId cmdId);
```

2.3.12. BRGetLastErrorMessage

The error information can be acquired by calling 'BRGetLastErrorMessage' once error appear.

```
BDR_API char* BRGetLastErrorMessage();
```

Return Values

Return the last error message.

Remarks

The error information can be acquired by calling 'BRGetLastErrorMessage' once error occurred during function callback.

3. Illustration

NeuronDataReader library supports the most popular developing environments such as C/C++/MFC、WPF/C#、Mac Cocoa, and game engine like Unity, Unigine etc.

Bellow will illustrations how library be used in C#.

For windows platform, start Visual Studio 2012, select”New…” in the start page.

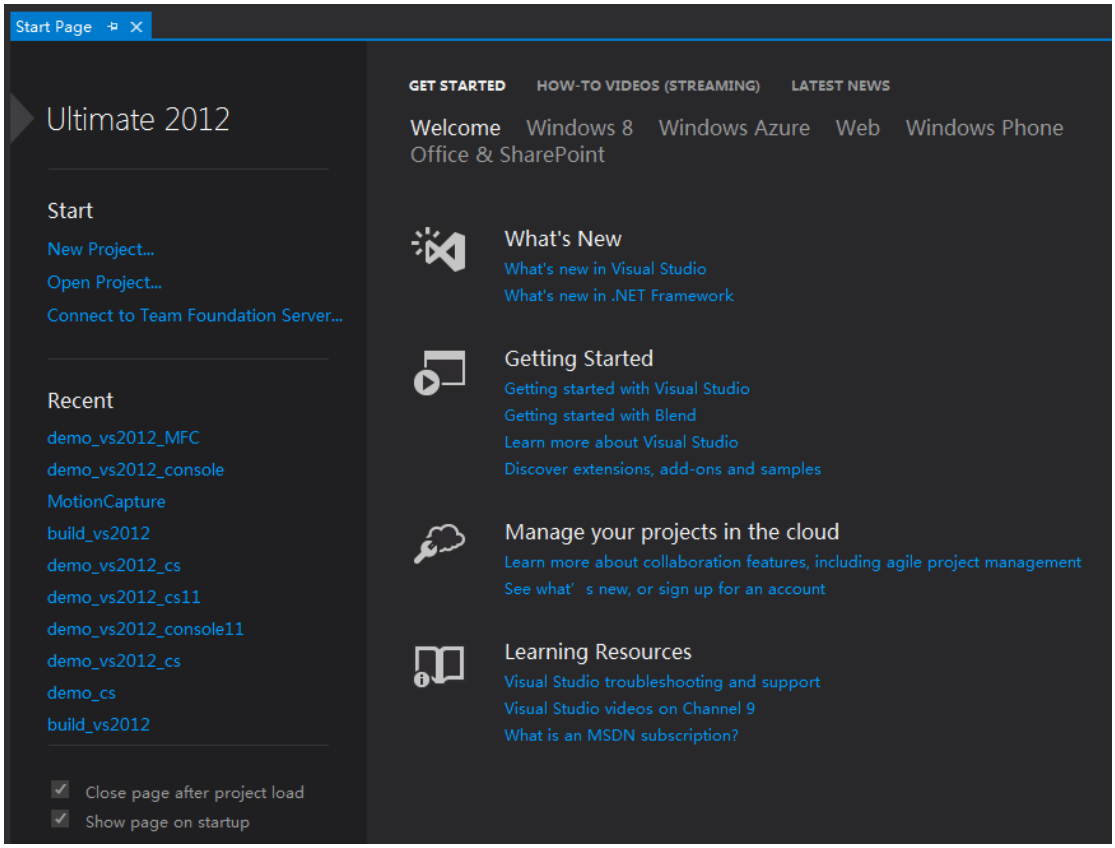


Fig. 3-1

Choose Visual C#--> WPF Application in the template list, name the project “demo_cs”:

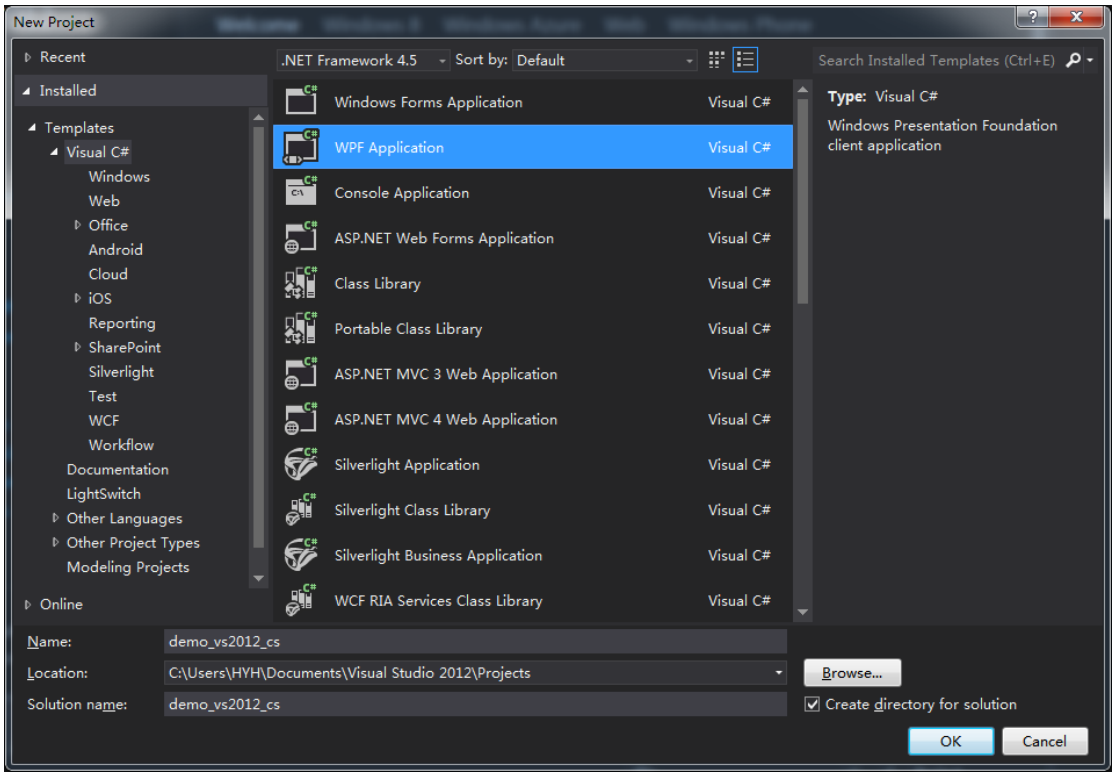


Fig. 3-2

Click OK, and an empty WPF project is created.

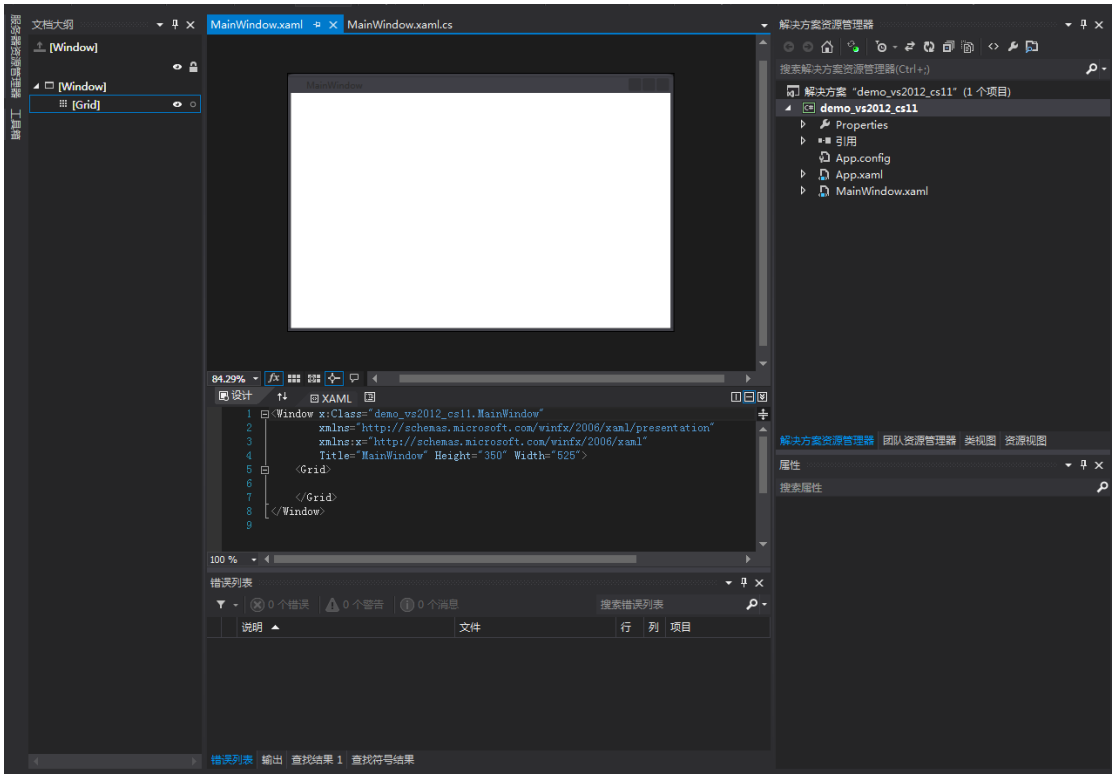


Fig. 3-3 WPF initial project

As the API in NeuronDataReader.dll cannot be accessed directly in C#, so a function import class need to be created to wrap the functions in library.

NeuronDataReader wrap class

```
/* Copyright: Copyright 2014 Beijing Noitom Technology Ltd. All Rights reserved.
* Pending Patents: PCT/CN2014/085659 PCT/CN2014/071006
*
* Licensed under the Neuron SDK License Beta Version (the “License");
* You may only use the Neuron SDK when in compliance with the License,
* which is provided at the time of installation or download, or which
* otherwise accompanies this software in the form of either an electronic or a hard copy.
*
* Unless required by applicable law or agreed to in writing, the Neuron SDK
* distributed under the License is provided on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing conditions and
* limitations under the License.
*/


using System;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;  // For DllImport()


namespace NeuronDataReaderWraper
{
    #region Basic data types
    /// <summary>
    /// Socket connection status
    /// </summary>
    public enum SocketStatus
    {
        CS_Running,
        CS_Starting,
        CS_OffWork,
    };

    /// <summary>
    /// Data version
    /// </summary>
    public struct DataVersion
    {
```

```
        public byte BuildNumb;         // Build number
        public byte Revision;          // Revision number
        public byte Minor;             // Subversion number
        public byte Major;             // Major version number
    };

    /// <summary>
    /// Header format of BVH data
    /// </summary>
    [StructLayout(LayoutKind.Sequential, Pack=1)]
    public struct BvhDataHeader
    {
        public ushort HeaderToken1;    // Package start token: 0xDDFF
        public DataVersion DataVersion;// Version of community data format. e.g.: 1.0.0.2
        public UInt32 DataCount;       // Values count, 180 for without disp data
        public UInt32 bWithDisp;       // With/out dispement
        public UInt32 bWithReference;  // With/out reference bone data at first
        public UInt32 AvatarIndex;     // Avatar index
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
        public string AvatarName;      // Avatar name
        public UInt32 Reserved1;       // Reserved, only enable this package has 64bytes
length
        public UInt32 Reserved2;       // Reserved, only enable this package has 64bytes
length
        public ushort HeaderToken2;    // Package end token: 0xEEFF
    };
    #endregion

    #region Command data types
    /// <summary>
    /// Command identitys
    /// </summary>
    public enum CmdId
    {
        Cmd_BoneSize,                 // Id used to request bone size from server
        Cmd_AvatarName,               // Id used to request avatar name from server
        Cmd_FaceDirection,            // Id used to request face direction from server
        Cmd_DataFrequency,            // Id used to request data sampling frequency from
server
        Cmd_BvhInheritance,            // Id used to request bvh inheritance from server
        Cmd_AvatarCount,           // Id used to request avatar count from server
        Cmd_CombinationMode,          //
        Cmd_RegisterEvent,            //
        Cmd_SetAvatarName,            //
```

```
    };

    // Sensor binding combination mode
    public enum SensorCombinationModes
    {
        SC_ArmOnly,             // Left arm or right arm only
        SC_UpperBody,           // Upper body, include one arm or both arm, must have chest
node
        SC_FullBody,            // Full body mode
    };

    /// <summary>
    /// Header format of Command returned from server
    /// </summary>
    [StructLayout(LayoutKind.Sequential, Pack=1)]
    public struct CommandPack
    {
        public UInt16 Token1;                 // Command start token: 0xAAFF
        public UInt32 DataVersion;            // Version of community data format. e.g.:
1.0.0.2
        public UInt32 DataLength;             // Package length of command data, by byte.
        public UInt32 DataCount;              // Count in data array, related to the specific
command.
        public CmdId CommandId;               // Identity of command.
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 40)]
        public byte[] CmdParaments;           // Command paraments
        public UInt32 Reserved1;              // Reserved, only enable this package has
32bytes length. Maybe used in the future.
        public UInt16 Token2;                 // Package end token: 0xBBFF
    };


    /// <summary>
    /// Fetched bone size from server
    /// </summary>
    [StructLayout(LayoutKind.Sequential, Pack=1)]
    public struct CmdResponseBoneSize
    {
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 60)]
        public string BoneName;       // Bone name
        public float BoneLength;      // Bone length
    };

    #endregion
```

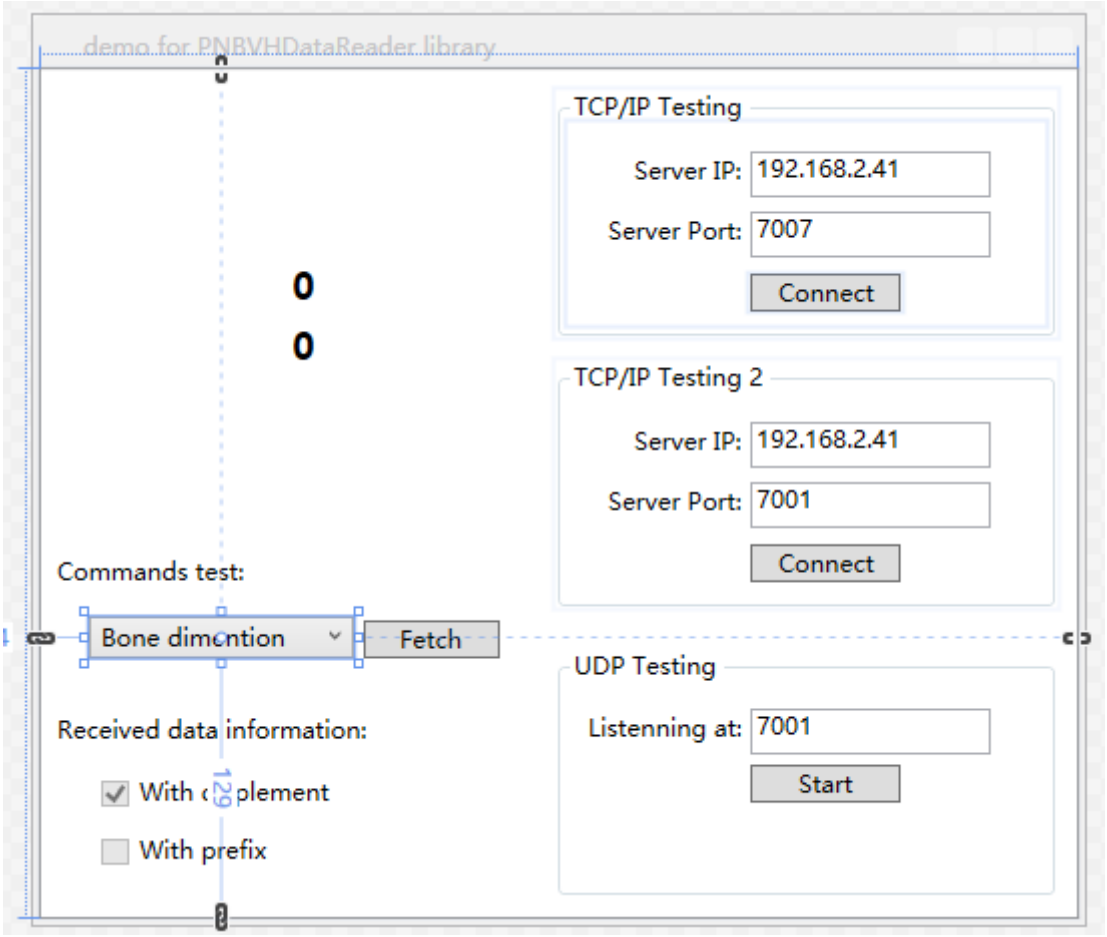
```
    #region Callbacks for data output
    /// <summary>
    /// FrameDataReceived CALLBACK
    /// Remarks
    ///   The related information of the data stream can be obtained from BvhDataHeader.
    /// </summary>
    /// <param name="customObject">User defined object.</param>
    /// <param name="sockRef">Connector reference of TCP/IP client as identity.</param>
    /// <param name="bvhDataHeader">A BvhDataHeader type pointer, to output the BVH data
format information.</param>
    /// <param name="data">Float type array pointer, to output binary data.</param>
    [UnmanagedFunctionPointer(CallingConvention.StdCall)]
    public delegate void FrameDataReceived(IntPtr customObject, IntPtr sockRef, IntPtr
bvhDataHeader, IntPtr data);

    /// <summary>
    /// Callback for command communication data with TCP/IP server
    /// </summary>
    /// <param name="customedObj">User defined object.</param>
    /// <param name="sockRef">Connector reference of TCP/IP client as identity.</param>
    /// <param name="cmdHeader">A CommandHeader type pointer contains command data
information.</param>
    /// <param name="cmdData">Data pointer of command, related to the specific
command.</param>
    /// <remark>The related information of the command data can be obtained from
CommandHeader. The data content is identified by its command id.</remark>
    [UnmanagedFunctionPointer(CallingConvention.StdCall)]
    public delegate void CommandDataReceived(IntPtr customedObj, IntPtr sockRef, IntPtr
cmdHeader, IntPtr cmdData);

    /// <summary>
    /// SocketStatusChanged CALLBACK
    /// Remarks
    ///   As convenient, use BRGetSocketStatus() to get status manually other than register
this callback
    /// </summary>
    /// <param name="customObject">User defined object.</param>
    /// <param name="sockRef">Socket reference of TCP or UDP service identity.</param>
    /// <param name="bvhDataHeader">Socket connection status</param>
    /// <param name="data">Socket status description.</param>
    [UnmanagedFunctionPointer(CallingConvention.StdCall)]
    public delegate void SocketStatusChanged(IntPtr customObject, IntPtr sockRef,
SocketStatus status, [MarshalAs(UnmanagedType.LPStr)]string msg);
```

```
    #endregion

    // API exportor
    public class NeuronDataReader
    {
        #region Importor definition
#if UNITY_IPHONE && !UNITY_EDITOR
        private const string ReaderImportor = "__Internal";
#elif _WINDOWS
        private const string ReaderImportor = "NeuronDataReader.dll";
#else
        private const string ReaderImportor = "NeuronDataReader";
#endif
        #endregion

        #region Functions API
        /// <summary>
        /// Register receiving and parsed frame data callback
        /// </summary>
        /// <param name="customedObj">Client defined object. Can be null</param>
        /// <param name="handle">Client defined function.</param>
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern void BRRegisterFrameDataCallback(IntPtr customedObj,
FrameDataReceived handle);

        /// <summary>
        ///
        /// </summary>
        /// <returns></returns>
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        //[return: MarshalAs(UnmanagedType.LPStr)]
        private static extern IntPtr BRGetLastErrorMessage();
        /// <summary>
        /// Call this function to get what error occured in library.
        /// </summary>
        /// <returns></returns>
        public static string strBRGetLastErrorMessage()
        {
            // Get message pointer
            IntPtr ptr = BRGetLastErrorMessage();
            // Construct a string from the pointer.
            return Marshal.PtrToStringAnsi(ptr);
```

```
        }

        // Register TCP socket status callback
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern void BRRegisterSocketStatusCallback(IntPtr customedObj,
SocketStatusChanged handle);

        // Connect to server by TCP/IP
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern IntPtr BRConnectTo(string serverIP, int nPort);

        // Check TCP/UDP service status
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern SocketStatus BRGetSocketStatus(IntPtr sockRef);

        // Close a TCP/UDP service
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern void BRCloseSocket(IntPtr sockRef);

        // Start a UDP service to receive data at 'nPort'
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern IntPtr BRStartUDPServiceAt(int nPort);
        #endregion

        #region Commands API
        /// <summary>
        /// Register receiving and parsed Cmd data callback
        /// </summary>
        /// <param name="customedObj">Client defined object. Can be null</param>
        /// <param name="handle">Client defined function.</param>
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern void BRRegisterCommandDataCallback(IntPtr customedObj,
CommandDataReceived handle);

        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern bool BRRegisterAutoSyncParmeter(IntPtr sockRef, CmdId cmdId);
```

```
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern bool BRUnregisterAutoSyncParmeter(IntPtr sockRef, CmdId cmdId);

        // Check TCP connect status
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern bool BRCommandFetchAvatarDataFromServer(IntPtr sockRef, int
avatarIndex, CmdId cmdId);

        // Check TCP connect status
        [DllImport(ReaderImportor, CallingConvention = CallingConvention.Cdecl, CharSet =
CharSet.Ansi)]
        public static extern bool BRCommandFetchDataFromServer(IntPtr sockRef, CmdId cmdId);
        #endregion
    }
}
```

Then add some controllers in the main window, to connect to the server and display the basic information of the received BVH data:



Fir. 3-4 Interface configuration

Copy the NeuronDataReader.dll to the same folder of “*.exe” file, and then run the program. The result is shown below:

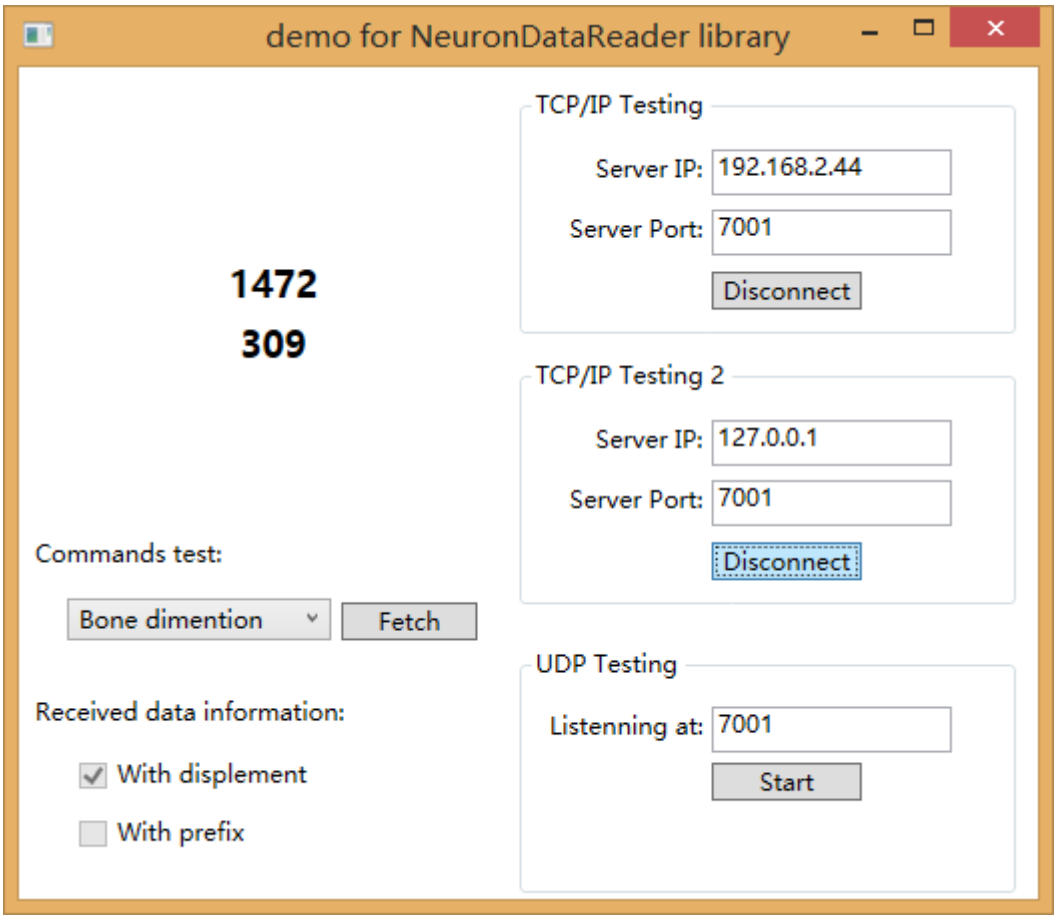


Fig. 3-5 Running Result

4. Known Bugs

If any bug or issues not figured out in this document, please report to me at: yuanhui.he@noitom.com

Thank you!

Appendix A: Skeleton data sequence in array

	Bone Name	Sequence In Data Block
Body	Hips (Position)	0
	Hips	1
	RightUpLeg	2
	RightLeg	3
	RightFoot	4
	LeftUpLeg	5
	LeftLeg	6
	LeftFoot	7
	Spine	8
	Spine1	9
	Spine2	10
	Spine3	11
	Neck	12
	Head	13
	RightShoulder	14
	RightArm	15
	RightForeArm	16
	RightHand	17
Fingers	RightHandThumb1	18
	RightHandThumb2	19
	RightHandThumb3	20
	RightInHandIndex	21
	RightHandIndex1	22
	RightHandIndex2	23
	RightHandIndex3	24
	RightInHandMiddle	25
	RightHandMiddle1	26
	RightHandMiddle2	27
	RightHandMiddle3	28
	RightInHandRing	29
	RightHandRing1	30
	RightHandRing2	31
	RightHandRing3	32
	RightInHandPinky	33
	RightHandPinky1	34
	RightHandPinky2	35
	RightHandPinky3	36
Body	LeftShoulder	37
	LeftArm	38
	LeftForeArm	39
	LeftHand	40
Fingers	LeftHandThumb1	41
	LeftHandThumb2	42
	LeftHandThumb3	43
	LeftInHandIndex	44
	LeftHandIndex1	45
	LeftHandIndex2	46
	LeftHandIndex3	47
	LeftInHandMiddle	48
	LeftHandMiddle1	49
	LeftHandMiddle2	50
	LeftHandMiddle3	51
	LeftInHandRing	52
	LeftHandRing1	53
	LeftHandRing2	54
	LeftHandRing3	55
	LeftInHandPinky	56
	LeftHandPinky1	57
	LeftHandPinky2	58
	LeftHandPinky3	59

Appendix B: BVH header template

```
HIERARCHY
ROOT Hips
{
	OFFSET 0.00 104.19 0.00
	CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
	JOINT RightUpLeg
	{
		OFFSET -11.50 0.00 0.00
		CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
		JOINT RightLeg
		{
			OFFSET 0.00 -48.00 0.00
			CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
			JOINT RightFoot
			{
				OFFSET 0.00 -48.00 0.00
				CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
				End Site
				{
					OFFSET 0.00 -1.81 18.06
				}
			}
		}
	}
	JOINT LeftUpLeg
	{
		OFFSET 11.50 0.00 0.00
		CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
		JOINT LeftLeg
		{
			OFFSET 0.00 -48.00 0.00
			CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
			JOINT LeftFoot
			{
				OFFSET 0.00 -48.00 0.00
				CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
				End Site
				{
					OFFSET 0.00 -1.81 18.06
				}
			}
		}
```

```
}
JOINT Spine
{
	OFFSET 0.00 13.88 0.00
	CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
	JOINT Spine1
	{
		OFFSET 0.00 11.31 0.00
		CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
		JOINT Spine2
		{
			OFFSET 0.00 11.78 0.00
			CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
			JOINT Spine3
			{
				OFFSET 0.00 11.31 0.00
				CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
				 JOINT Neck
				{
					OFFSET 0.00 12.09 0.00
					CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
					JOINT Head
					{
						OFFSET 0.00 9.00 0.00
						CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
						End Site
						{
							OFFSET 0.00 18.00 0.00
						}
					}
				}
				JOINT RightShoulder
				 {
					OFFSET -3.50 8.06 0.00
					CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
					JOINT RightArm
					{
						OFFSET -17.50 0.00 0.00
						CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
						JOINT RightForeArm
						{
							OFFSET -29.00 0.00 0.00
							CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
							JOINT RightHand
```

```
{
    OFFSET -28.00 0.00 0.00
    CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
    JOINT RightHandThumb1
    {
        OFFSET -2.70 0.21 3.39
        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
        JOINT RightHandThumb2
        {
            OFFSET -2.75 -0.64 2.83
            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
            JOINT RightHandThumb3
            {
                OFFSET -2.13 -0.81 1.59
                CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                End Site
                {
                    OFFSET -1.80 -0.90 1.80
                }
            }
        }
    }
    JOINT RightInHandIndex
    {
        OFFSET -3.50 0.55 2.15
        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
        JOINT RightHandIndex1
        {
            OFFSET -5.67 -0.10 1.09
            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
            JOINT RightHandIndex2
            {
                OFFSET -3.92 -0.19 0.20
                CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                JOINT RightHandIndex3
                {
                    OFFSET -2.22 -0.14 -0.08
                    CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                    End Site
                    {
                        OFFSET -2.28 0.00 0.00
                    }
                }
            }
```

```
        }
    }
    JOINT RightInHandMiddle
    {
        OFFSET -3.67 0.56 0.82
        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
        JOINT RightHandMiddle1
        {
            OFFSET -5.62 -0.09 0.34
            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
            JOINT RightHandMiddle2
            {
                OFFSET -4.27 -0.29 -0.20
                CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                JOINT RightHandMiddle3
                {
                    OFFSET -2.67 -0.21 -0.24
                    CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                    End Site
                    {
                        OFFSET -2.28 0.00 0.00
                    }
                }
            }
        }
    }
    JOINT RightInHandRing
    {
        OFFSET -3.65 0.59 -0.14
        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
        JOINT RightHandRing1
        {
            OFFSET -5.00 -0.02 -0.52
            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
            JOINT RightHandRing2
            {
                OFFSET -3.65 -0.29 -0.74
                CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                JOINT RightHandRing3
                {
                    OFFSET -2.55 -0.19 -0.44
                    CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                    End Site
                    {
```

```
                                        OFFSET -2.16 0.00 0.00
                                    }
                                }
                            }
                        }
                    }
                    JOINT RightInHandPinky
                    {
                        OFFSET -3.43 0.51 -1.30
                        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                        JOINT RightHandPinky1
                        {
                            OFFSET -4.49 -0.02 -1.18
                            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                            JOINT RightHandPinky2
                            {
                                OFFSET -2.85 -0.16 -0.90
                                CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                                JOINT RightHandPinky3
                                {
                                    OFFSET -1.77 -0.14 -0.66
                                    CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                                    End Site
                                    {
                                        OFFSET -1.68 0.00 0.00
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    JOINT LeftShoulder
    {
        OFFSET 3.50 8.06 0.00
        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
        JOINT LeftArm
        {
            OFFSET 17.50 0.00 0.00
            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
            JOINT LeftForeArm
            {
```

```
OFFSET 29.00 0.00 0.00
CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
JOINT LeftHand
{
   OFFSET 28.00 0.00 0.00
   CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
   JOINT LeftHandThumb1
   {
         OFFSET 2.70 0.21 3.39
         CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
         JOINT LeftHandThumb2
         {
               OFFSET 2.75 -0.64 2.83
               CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
               JOINT LeftHandThumb3
               {
                     OFFSET 2.13 -0.81 1.59
                     CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                     End Site
                     {
                           OFFSET 1.80 -0.90 1.80
                     }
               }
         }
   }
   JOINT LeftInHandIndex
   {
         OFFSET 3.50 0.55 2.15
         CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
         JOINT LeftHandIndex1
         {
               OFFSET 5.67 -0.10 1.08
               CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
               JOINT LeftHandIndex2
               {
                     OFFSET 3.92 -0.19 0.20
                     CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                     JOINT LeftHandIndex3
                     {
                           OFFSET 2.22 -0.14 -0.08
                           CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                           End Site
                           {
                                 OFFSET 2.28 0.00 0.00
```

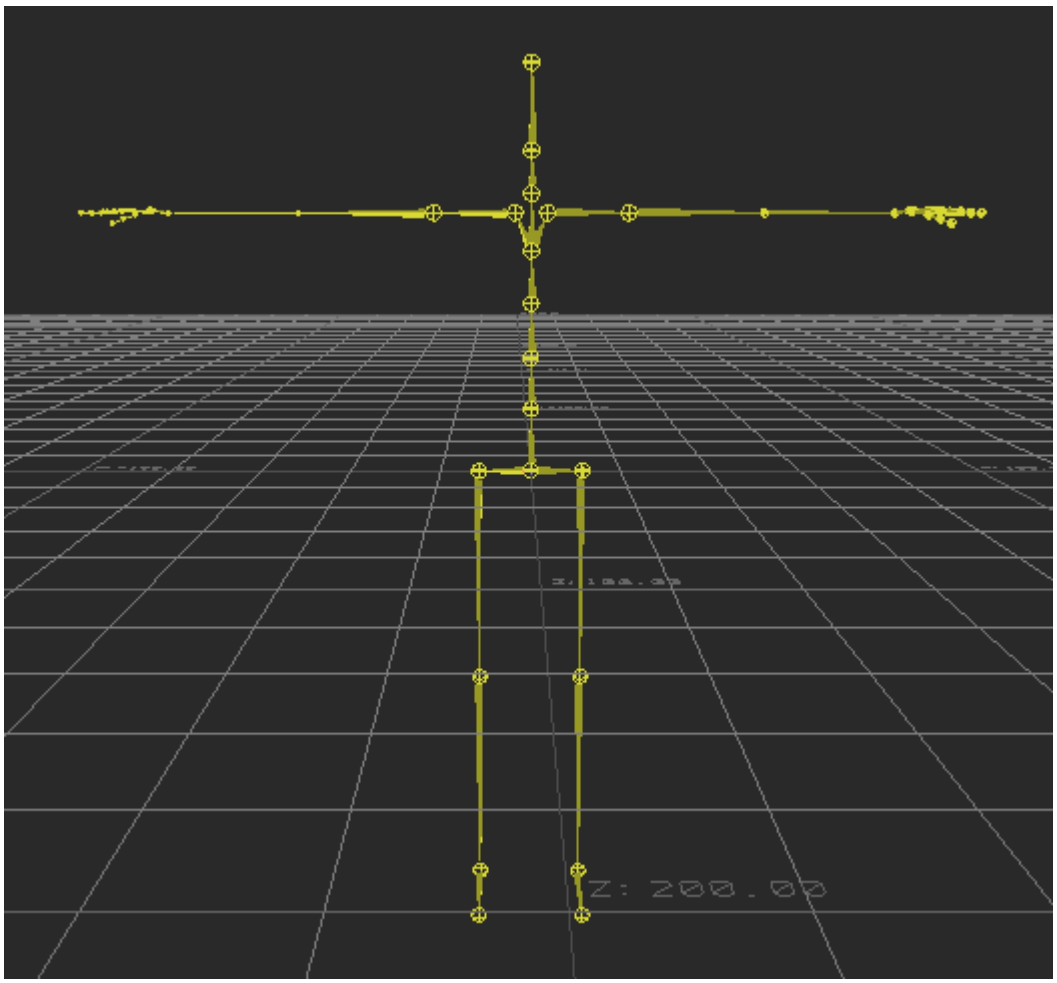
```
                    }
                }
            }
        }
    }
    JOINT LeftInHandMiddle
    {
        OFFSET 3.67 0.56 0.82
        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
        JOINT LeftHandMiddle1
        {
            OFFSET 5.62 -0.09 0.34
            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
            JOINT LeftHandMiddle2
            {
                OFFSET 4.27 -0.29 -0.20
                CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                JOINT LeftHandMiddle3
                {
                    OFFSET 2.67 -0.21 -0.24
                    CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                    End Site
                    {
                        OFFSET 2.28 0.00 0.00
                    }
                }
            }
        }
    }
    JOINT LeftInHandRing
    {
        OFFSET 3.65 0.59 -0.14
        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
        JOINT LeftHandRing1
        {
            OFFSET 5.00 -0.02 -0.52
            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
            JOINT LeftHandRing2
            {
                OFFSET 3.65 -0.29 -0.74
                CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                JOINT LeftHandRing3
                {
                    OFFSET 2.55 -0.19 -0.44
```

```
                                        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                                        End Site
                                        {
                                            OFFSET 2.16 0.00 0.00
                                        }
                                    }
                                }
                            }
                            JOINT LeftInHandPinky
                            {
                                OFFSET 3.43 0.51 -1.30
                                CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                                JOINT LeftHandPinky1
                                {
                                    OFFSET 4.49 -0.02 -1.18
                                    CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                                    JOINT LeftHandPinky2
                                    {
                                        OFFSET 2.85 -0.16 -0.90
                                        CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                                        JOINT LeftHandPinky3
                                        {
                                            OFFSET 1.77 -0.14 -0.66
                                            CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
                                            End Site
                                            {
                                                OFFSET 1.68 0.00 0.00
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
MOTION
```

Frames: 0

Frame Time: 0.010

Appendix C: Skeleton graph



Revision history

Revision	Author	date	Description/changes
D1	Yuanhui He	12/22/2014	Initial released
D2	Peng Gao	12/22/2014	**Added:** Description of APIs. **Modified:** Format edit.
D3	Siyuan Deng	12/23/2014	**Added:** English translation. **Modified:**
D4	Yuanhui He	12/25/2014	**Added:** **Modified:** Delete string data stream type.
D5	Jinzhou Chen	12/25/2014	**Modified:** Modify the English translation.
D6	Yuanhui He	12/25/2014	**Modified:** Modify the English translation and some API description.
D7	Yuanhui He Siyuan Deng	1/26/2015	**Added:** Appendix, Skeleton Data format **Modified:** Data format description
D8	Yuanhui He	2/3/2015	**Added:** UDP protocol type support.
D9	Tobi	11/3/2015	**Modify:** English review.
D10	Yuanhui He	20/3/2015	**Add:** Multi-client is supported. **Modify:** English review.
D11	Yuanhui He	24/4/2015	**Add:** Some commands or APIs added to be used to sync parameters or data from server. **Delete:** Unity demo
D12	Yuanhui He	5/5/2015	**Modify:** Merged some TCP/UDP functions.