Neuron Data Reader Runtime API

Documentation

By Yuanhui He
Xudong Wang

16.08.2018

NeuronDataReader b18

Noitom Technology Co., Ltd.

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

Contents

1. Overview

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.

1.1. NeuronDataReader framework

The structure of NeuronDataReader library is shown below.

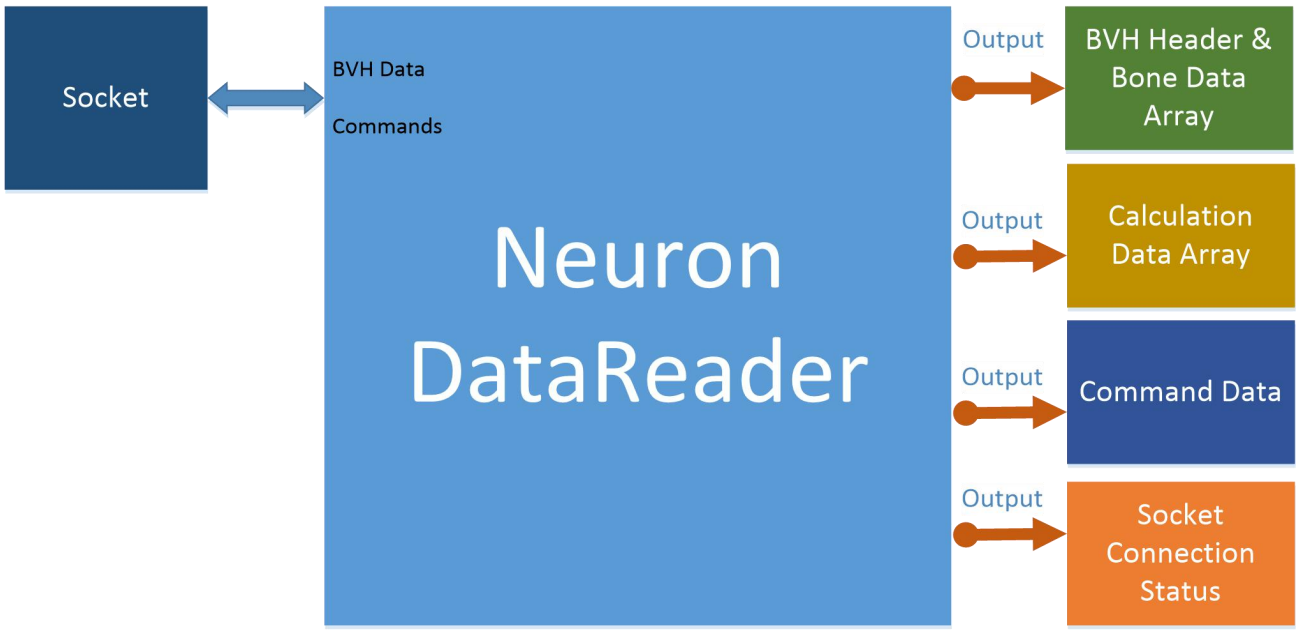


Fig. 1-1 NeuronDataReader Overview

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

1.2. Skeleton Data Format

In this version, skeleton data include BVH data,calculation data and some command. And the skeleton data is output by the `Callbacks as the 2.2` .

1.2.1. BVH Data

The action data, with BVH format, is output by callback. The data stream of every fame includes the BVH header and BVH motion data of float type. All information of the skeleton data, such as prefix, displacements settings, etc. are included in a BvhDataHeaderEx 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.

Through the Network, NeuronDataReader receives BVH data frames from Axis Neuron, and BVH data in each frame include all the motion data of 59 bones.

For the BVH data with displacement, the data of each bone has 6 float: 3 displacements(X Y Z) and 3 rotation data (Default rotation order is Y X Z).

For the BVH data without displacement, except the root node (Hip) having displacement and rotation, other bones only have rotation data.

So, if users want to get the information (position or pose) of the specified bone, they could calculate the relevant numerical index according to the following formula.

1) BVH data with displacement

Displacement_X = bone index * 6 + 0

Displacement_Y = bone index * 6 + 1

Displacement_Z = bone index * 6 + 2

Rotation_Y = bone index * 6 + 3

Rotation_X = bone index * 6 + 4

Rotation_Z = bone index * 6 + 5

2) BVH data without displacement

Except rotation, only the hip node has displacement data.

Root_Displacement_X = 0

Root_Displacement_Y = 1

Root_Displacement_Z = 2

Rotation_Y = 3 + bone index * 3 + 0

Rotation_X = 3 + bone index * 3 + 1

Rotation_Z = 3 + bone index * 3 + 2

3) Introduce of BVH coordinate system

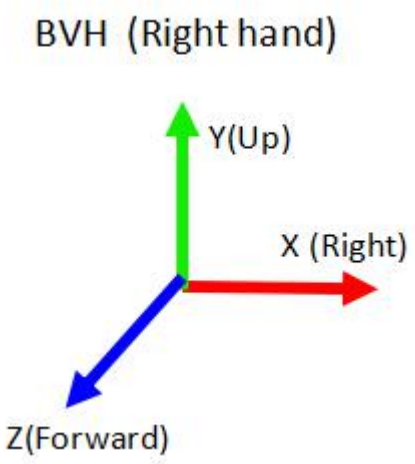


Fig. 1-2 BVH coordinate

4) BVH data with Reference

In order to translate or rotate the whole skeleton and do not change the data of bones in the skeleton, a new node, regarded as the parent node of the root node, could be added in the BVH data structure. As a result, as long as change the displacement of the new node can translate the whole skeleton model, and changing the pose of the new node can rotate the whole skeleton model. This new node is defined as Reference.

There is displacement data and pose data in the Reference, totally 6 value.

Therefore, for the above BVH data, if the data includes Reference, the index number of the skeleton data is obtained by adding 6 as the offset.

The structure of BVH data could be found in Appendix A.

NOTE:The pose data of each node is relative to the parent node, and the pose of the root node "Hips" is relative to the reference point.The displacement of the BVH data is a constant usually, it reflects the offset of the initial position of the skeleton relative to the initial position of the parent node of the TPose. So it is a constant.

1.2.2.Calculation Data

Through the Network, NeuronDataReader receives calculation data frames from Axis Neuron, and calculation data in each frame include all the sensor data and motion data of 59 bones, and at last also include the contact state of two feet.

For the calculation data, the data of each bone has its position(X Y Z, with global coordinate), velocity(X Y Z with global coordinate), sensor quaternion(W X Y Z with global coordinate), sensor accelerated velocity (X Y Z with modules' coordinate), and gyro(X Y Z with modules' coordinate), totally 16 float data.

The calculation data format is shown as below:

Position_X = bone index * 16 + 0
Position_Y = bone index * 16 + 1
Position_Z = bone index * 16 + 2

Velocity_X = bone index * 16 + 3
Velocity_Y = bone index * 16 + 4
Velocity_Z = bone index * 16 + 5

Quaternion_W = bone index * 16 + 6
Quaternion_X = bone index * 16 + 7
Quaternion_Y = bone index * 16 + 8
Quaternion_Z = bone index * 16 + 9

Accelerated velocity_X = bone index * 16 + 10
Accelerated velocity_Y = bone index * 16 + 11
Accelerated velocity_Z = bone index * 16 + 12

Gyro_X = bone index * 16 + 13
Gyro_Y = bone index * 16 + 14
Gyro_Z = bone index * 16 + 15

The output order in each frame data is No.1 to No.21 before the version b15,and the continue is the flag of left and right foot(1-touched ground, 0-untouched ground). After the b15, Bone data of both hands added. The bone index or output order for calculation data could be found in Appendix C and Appendix D.

1.3.Data Frequency

The frequency of data output from NeuronDataReader depends on the number of sensors worn by the current user. We set, when the number of nodes in the device is less than 18, the corresponding acquisition frequency is 120Hz; when the number of nodes is not less than 18, the acquisition frequency is 60Hz. Correspondingly, the call frequency of the callback function is the same as the data acquisition frequency.

It should be noted that in the process of data transmission by network, there is a very small probability of losing the frame. So, although the frequency is certain, the number of data received by NeuronDataReader maybe change.

1.4. 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.

2. Reference

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

2.1. Data type definitions

2.1.1.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;
```

2.1.2.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 DataVersion
{
    uint32_t _VersionMask;
    struct
    {
        uint8_t BuildNumb;      // Build number
        uint8_t Revision;       // Revision number
        uint8_t Minor;          // Subversion number
        uint8_t Major;          // Major version number
    };
}DATA_VER;
```

2.1.3.Header of BVH data stream

```
// Header format of BVH data
typedef struct _BvhDataHeader
{
    uint16_t   Token1;          // Package start token: 0xDDFF
    DATA_VER   DataVersion;     // Version of community data format. e.g.: 1.0.0.2
    uint16_t   DataCount;       // Values count
    uint8_t    WithDisp;        // With/out displacement
```

```
    uint8_t    WithReference;    // With/out reference bone data at first
    uint32_t   AvatarIndex;      // Avatar index
    uint8_t    AvatarName[32];   // Avatar name
    uint32_t   FrameIndex;       // Frame data index
    uint32_t   Reserved;         // Reserved, only enable this package has 64bytes length
    uint32_t   Reserved1;        // Reserved, only enable this package has 64bytes length
    uint32_t   Reserved2;        // Reserved, only enable this package has 64bytes length
    uint16_t   Token2;           // Package end token: 0xEEFF
}BvhDataHeader;
```

For details, please refer to 1.2.1.

2.1.4.Header of Calculation data stream

```
// Header format of BVH data
typedef struct _CalcDataHeader
{
    uint16_t   Token1;          // Package start token: 0x88FF
    DATA_VER   DataVersion;     // Version of community data format. e.g.: 1.0.0.3
    uint32_t   DataCount;       // Values count
    uint32_t   AvatarIndex;     // Avatar index
    uint8_t    AvatarName[32];  // Avatar name
    uint32_t   FrameIndex;      // Frame data index
    uint32_t   Reserved1;       // Reserved, only enable this package has 64bytes length
    uint32_t   Reserved2;       // Reserved, only enable this package has 64bytes length
    uint32_t   Reserved3;       // Reserved, only enable this package has 64bytes length
    uint16_t   Token2;          // Package end token: 0x99FF
}CalcDataHeader;
```

For details, please refer to 1.2.2.

2.1.5. CommandIdentity

```
// Support command
typedef enum _CommandIdentity
{
    Cmd_ZeroOutPosition = 1001,     // zero out
    Cmd_ZeroOutAllAvatar = 1002,    // zero out all
    Cmd_BvhRotation = 1003,         // bvh rotation
    Cmd_BoneSize = 1004,            // bone size
    Cmd_BvhHeader = 1005,           // bvh header
}CmdId;
```

2.1.6.Definition of command

```
typedef struct
{
    CommandPackHeader cmdPackHeader;
    union
    {
```

```
        unsigned char cData[MAX_PACKETSIZE];
        char          szData[MAX_PACKETSIZE];
        long          lData[MAX_PACKETSIZE / 4];
        unsigned long ulData[MAX_PACKETSIZE / 4];
        float         fData[MAX_PACKETSIZE / 4];
    } Data;                                 // Payload
} CommandPack;
```

2.1.7.Definition of the command’s header

```
// Header format of Cmd data
typedef struct
{
    uint16_t Token1;           // Command start token: 0xAAFF
    uint16_t CommandType;
    uint16_t CommandId;
    uint32_t nDataBytes;       // Num bytes in payload
    uint32_t nReserved;        // Make this struct to align to 8 bytes
    uint16_t Token2;           // Package end token: 0xBBFF
}CommandPackHeader;
```

2.1.8.Bone dimensions

```
// Bone dimensions, unit: meter
typedef struct _BoneDimension
{
    float Head;             // Bone length of head
    float Neck;             // Bone length of neck
    float Body;             // Length of body
    float ShoulderWidth;    // Width of shoulder
    float UpperArm;         // Bone length of upper arm
    float Forearm;          // Bone length of forearm
    float Palm;             // Bone length of hand
    float HipWidth;         // Width of hip
    float UpperLeg;         // Bone length of upper leg
    float LowerLeg;         // Bone length of lower leg
    float HeelHeight;       // Heel height
    float FootLength;       // Foot length
};
```

2.1.9.BVH rotate orders

```
// BVH rotate orders
typedef enum _RotateOrders
{
    RO_XZY,
    RO_YXZ,
    RO_XYZ,
    RO_YZX,
```

```
    RO_ZXY,
    RO_ZYX,
    RO_Unknown,             // Unknown type
}RotateOrders;
```

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.Calculation data callback

```
    typedef void (CALLBACK * CalculationDataReceived)(void* customedObj, SOCKET_REF sender, CalcDataHeader * header, float* data);
```

Parameters

customedObj

User defined object.

sender

Connector reference of TCP/IP client as identity.

Pack

CalcDataHeader type pointer, to output the calculation data format information.

data

Float type array pointer, to output binary data.

Remarks

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

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.BRRegisterCalculationDataCallback

Register the calculation data receiving callback handle:

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

Parameters

customedObj

User defined object.

handle

A function pointer of CalculationDataReceived type.

Remarks

The handle of CalculationDataReceived 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_REF BRConnectTo(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 referred socket.

2.3.8.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.

2.3.9.BRConnectCmd

A Command pipe to server can be created by calling he 'BRConnectCmd'.

```
// Create a cmd pipe to connect to server
NEURONDATAREADER_API SOCKET_REF BRConnectCmd(char* serverIP, int nPort);
```

Parameters

serverIP[in]

Server's IP address.

nPort[in]

Server's port.

Return Values

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

2.3.10. ZeroOut

Send command through the Command pipe created by the 'BRConnectCmd'.

```
// ZeroOut
NEURONDATAREADER_API bool ZeroOut(SOCKET_REF sockRef, int avatarIndex);
```

Parameters

sockRef[in]

a handle of connector [cmd].

avatarIndex[in]

avatar index in Axis.

Return Values

If successfully, return true; otherwise false is returned.

2.3.11. ZeroOutAll

Send command through the Command pipe created by the 'BRConnectCmd'.

```
// BRZeroOutAll
NEURONDATAREADER_API bool ZeroOutAll(SOCKET_REF sockRef);
```

Parameters

sockRef[in]

a handle of connector [cmd].

Return Values

If successfully, return true; otherwise false is returned.

2.3.12. GetBvhRotation

Send command through the Command pipe created by the 'BRConnectCmd'.

```
// GetBvhRotation
NEURONDATAREADER_API bool GetBvhRotation(SOCKET_REF sockRef, RotateOrders* order);
```

Parameters

sockRef[in]

a handle of connector [cmd].

order[out]

bvh rotate order.

Return Values

If successfully, return true and bvh rotate order; otherwise false is returned.

2.3.13. BRGetBoneSize

Send command through the Command pipe created by the 'BRConnectCmd'.

```
// BRGetBoneSize
NEURONDATAREADER_API bool GetBoneSize(SOCKET_REF sockRef, int sockRef, BoneDimension* dimensions);
```

Parameters

sockRef[in]

a handle of connector [cmd].

avatarIndex[in]

avatar index in Axis.

Dimensions[out]

Bone dimensions, unit: meter, refer to BoneDimension define.

Return Values

If successfully, return true and Bone dimensions; otherwise false is returned.

2.3.14. GetBvhHeader

Send command through the Command pipe created by the 'BRConnectCmd'.

```
// GetBvhHeader
NEURONDATAREADER_API bool GetBvhHeader(SOCKET_REF sockRef, int avatarIndex, char* data, int len);
```

Parameters

sockRef[in]

a handle of connector [cmd pipe].

avatarIndex[in]

avatar index in Axis.

data[out]

bvh header, refer to Appendix B.

len[in]

length of data.

Return Values

If successfully, return true; otherwise false is returned.

3. Known Bugs

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

Thank you!

Appendix A: Skeleton Data Sequence in Array

	Bone Name	Sequence In Data Block
Body	Hips	0
	RightUpLeg	1
	RightLeg	2
	RightFoot	3
	LeftUpLeg	4
	LeftLeg	5
	LeftFoot	6
	Spine	7
	Spine1	8
	Spine2	9
	Neck	10
	Neck1	11
	Head	12
	RightShoulder	13
	RightArm	14
	RightForeArm	15
	RightHand	16
Fingers	RightHandThumb1	17
	RightHandThumb2	18
	RightHandThumb3	19
	RightInHandIndex	20
	RightHandIndex1	21
	RightHandIndex2	22
	RightHandIndex3	23
	RightInHandMiddle	24
	RightHandMiddle1	25
	RightHandMiddle2	26
	RightHandMiddle3	27
	RightInHandRing	28
	RightHandRing1	29
	RightHandRing2	30
	RightHandRing3	31
	RightInHandPinky	32
	RightHandPinky1	33
	RightHandPinky2	34
	RightHandPinky3	35
Body	LeftShoulder	36
	LeftArm	37
	LeftForeArm	38
	LeftHand	39

Fingers	LeftHandThumb1	40
	LeftHandThumb2	41
	LeftHandThumb3	42
	LeftInHandIndex	43
	LeftHandIndex1	44
	LeftHandIndex2	45
	LeftHandIndex3	46
	LeftInHandMiddle	47
	LeftHandMiddle1	48
	LeftHandMiddle2	49
	LeftHandMiddle3	50
	LeftInHandRing	51
	LeftHandRing1	52
	LeftHandRing2	53
	LeftHandRing3	54
	LeftInHandPinky	55
	LeftHandPinky1	56
	LeftHandPinky2	57
	LeftHandPinky3	58

Appendix B: BVH Header Template

```
HIERARCHY
ROOT Hips
{
	OFFSET 0.000 84.102 0.000
	CHANNELS 6 Xposition Yposition Zposition Yrotation Xrotation Zrotation
	JOINT RightUpLeg
	{
		OFFSET -9.500 -0.000 0.000
		CHANNELS 3 Yrotation Xrotation Zrotation
		JOINT RightLeg
		{
			OFFSET 0.000 -37.051 0.000
			CHANNELS 3 Yrotation Xrotation Zrotation
			JOINT RightFoot
			{
				OFFSET 0.000 -37.051 0.000
				CHANNELS 3 Yrotation Xrotation Zrotation
				End Site
				{
					OFFSET 0.000 -10.000 15.750
				}
			}
		}
	}
	JOINT LeftUpLeg
	{
		OFFSET 9.500 -0.000 0.000
		CHANNELS 3 Yrotation Xrotation Zrotation
		JOINT LeftLeg
		{
			OFFSET 0.000 -37.051 0.000
			CHANNELS 3 Yrotation Xrotation Zrotation
			JOINT LeftFoot
			{
				OFFSET 0.000 -37.051 0.000
				CHANNELS 3 Yrotation Xrotation Zrotation
				End Site
				{
					OFFSET 0.000 -10.000 15.750
				}
			}
		}
```

```
}
JOINT Spine
{
	OFFSET 0.000 7.140 0.000
	CHANNELS 3 Yrotation Xrotation Zrotation
	JOINT Spine1
	{
		OFFSET 0.000 15.810 0.000
		CHANNELS 3 Yrotation Xrotation Zrotation
		JOINT Spine2
		{
			OFFSET 0.000 11.220 0.000
			CHANNELS 3 Yrotation Xrotation Zrotation
			JOINT Neck
			{
				OFFSET 0.000 16.830 0.000
				CHANNELS 3 Yrotation Xrotation Zrotation
				JOINT Neck1
				{
					OFFSET 0.000 4.500 0.000
					CHANNELS 3 Yrotation Xrotation Zrotation
					JOINT Head
					{
						OFFSET 0.000 4.500 0.000
						CHANNELS 3 Yrotation Xrotation Zrotation
						End Site
						{
							OFFSET 0.000 17.000 0.000
						}
					}
				}
			}
			JOINT RightShoulder
			{
				OFFSET -2.550 11.730 0.000
				CHANNELS 3 Yrotation Xrotation Zrotation
				JOINT RightArm
				{
					OFFSET -11.450 0.000 0.000
					CHANNELS 3 Yrotation Xrotation Zrotation
					JOINT RightForeArm
					{
						OFFSET -25.000 0.000 0.000
						CHANNELS 3 Yrotation Xrotation Zrotation
```

```
JOINT RightHand
{
	OFFSET -25.000 0.000 0.000
	CHANNELS 3 Yrotation Xrotation Zrotation
	JOINT RightHandThumb1
	{
		OFFSET -2.418 0.185 3.031
		CHANNELS 3 Yrotation Xrotation Zrotation
		JOINT RightHandThumb2
		{
			OFFSET -3.578 0.000 0.000
			CHANNELS 3 Yrotation Xrotation Zrotation
			JOINT RightHandThumb3
			{
				OFFSET -2.485 0.000 0.000
				CHANNELS 3 Yrotation Xrotation Zrotation
				End Site
				{
					OFFSET -2.131 0.000 0.000
				}
			}
		}
	}
	JOINT RightInHandIndex
	{
		OFFSET -3.132 0.494 1.922
		CHANNELS 3 Yrotation Xrotation Zrotation
		JOINT RightHandIndex1
		{
			OFFSET -5.068 -0.089 0.971
			CHANNELS 3 Yrotation Xrotation Zrotation
			JOINT RightHandIndex2
			{
				OFFSET -3.516 0.000 0.000
				CHANNELS 3 Yrotation Xrotation Zrotation
				JOINT RightHandIndex3
				{
					OFFSET -1.993 0.000 0.000
					CHANNELS 3 Yrotation Xrotation Zrotation
					End Site
					{
						OFFSET -1.754 0.000 0.000
					}
				}
```

```
            }
        }
    }
    JOINT RightInHandMiddle
    {
        OFFSET -3.285 0.502 0.735
        CHANNELS 3 Yrotation Xrotation Zrotation
        JOINT RightHandMiddle1
        {
            OFFSET -5.026 -0.082 0.305
            CHANNELS 3 Yrotation Xrotation Zrotation
            JOINT RightHandMiddle2
            {
                OFFSET -3.837 0.000 0.000
                CHANNELS 3 Yrotation Xrotation Zrotation
                JOINT RightHandMiddle3
                {
                    OFFSET -2.405 0.000 0.000
                    CHANNELS 3 Yrotation Xrotation Zrotation
                    End Site
                    {
                        OFFSET -1.918 0.000 0.000
                    }
                }
            }
        }
    }
    JOINT RightInHandRing
    {
        OFFSET -3.269 0.523 -0.125
        CHANNELS 3 Yrotation Xrotation Zrotation
        JOINT RightHandRing1
        {
            OFFSET -4.502 -0.021 -0.465
            CHANNELS 3 Yrotation Xrotation Zrotation
            JOINT RightHandRing2
            {
                OFFSET -3.344 0.000 0.000
                CHANNELS 3 Yrotation Xrotation Zrotation
                JOINT RightHandRing3
                {
                    OFFSET -2.320 0.000 0.000
                    CHANNELS 3 Yrotation Xrotation Zrotation
                    End Site
```

```
                                        {
                                            OFFSET -1.804 0.000 0.000
                                        }
                                    }
                                }
                            }
                        }
                        JOINT RightInHandPinky
                        {
                            OFFSET -3.071 0.456 -1.167
                            CHANNELS 3 Yrotation Xrotation Zrotation
                            JOINT RightHandPinky1
                            {
                                OFFSET -4.023 -0.021 -1.059
                                CHANNELS 3 Yrotation Xrotation Zrotation
                                JOINT RightHandPinky2
                                {
                                    OFFSET -2.678 0.000 0.000
                                    CHANNELS 3 Yrotation Xrotation Zrotation
                                    JOINT RightHandPinky3
                                    {
                                        OFFSET -1.692 0.000 0.000
                                        CHANNELS 3 Yrotation Xrotation Zrotation
                                        End Site
                                        {
                                            OFFSET -1.598 0.000 0.000
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
JOINT LeftShoulder
{
    OFFSET 2.550 11.730 0.000
    CHANNELS 3 Yrotation Xrotation Zrotation
    JOINT LeftArm
    {
        OFFSET 11.450 0.000 0.000
        CHANNELS 3 Yrotation Xrotation Zrotation
        JOINT LeftForeArm
```

```
{
    OFFSET 25.000 0.000 0.000
    CHANNELS 3 Yrotation Xrotation Zrotation
    JOINT LeftHand
    {
        OFFSET 25.000 0.000 0.000
        CHANNELS 3 Yrotation Xrotation Zrotation
        JOINT LeftHandThumb1
        {
            OFFSET 2.418 0.185 3.031
            CHANNELS 3 Yrotation Xrotation Zrotation
            JOINT LeftHandThumb2
            {
                OFFSET 3.578 0.000 0.000
                CHANNELS 3 Yrotation Xrotation Zrotation
                JOINT LeftHandThumb3
                {
                    OFFSET 2.485 0.000 0.000
                    CHANNELS 3 Yrotation Xrotation Zrotation
                    End Site
                    {
                        OFFSET 2.131 0.000 0.000
                    }
                }
            }
        }
        JOINT LeftInHandIndex
        {
            OFFSET 3.132 0.494 1.922
            CHANNELS 3 Yrotation Xrotation Zrotation
            JOINT LeftHandIndex1
            {
                OFFSET 5.068 -0.089 0.971
                CHANNELS 3 Yrotation Xrotation Zrotation
                JOINT LeftHandIndex2
                {
                    OFFSET 3.516 0.000 0.000
                    CHANNELS 3 Yrotation Xrotation Zrotation
                    JOINT LeftHandIndex3
                    {
                        OFFSET 1.993 0.000 0.000
                        CHANNELS 3 Yrotation Xrotation Zrotation
                        End Site
                        {
```

```
                    OFFSET 1.754 0.000 0.000
                }
            }
        }
    }
}
JOINT LeftInHandMiddle
{
    OFFSET 3.285 0.502 0.735
    CHANNELS 3 Yrotation Xrotation Zrotation
    JOINT LeftHandMiddle1
    {
        OFFSET 5.026 -0.082 0.305
        CHANNELS 3 Yrotation Xrotation Zrotation
        JOINT LeftHandMiddle2
        {
            OFFSET 3.837 0.000 0.000
            CHANNELS 3 Yrotation Xrotation Zrotation
            JOINT LeftHandMiddle3
            {
                OFFSET 2.405 0.000 0.000
                CHANNELS 3 Yrotation Xrotation Zrotation
                End Site
                {
                    OFFSET 1.918 0.000 0.000
                }
            }
        }
    }
}
JOINT LeftInHandRing
{
    OFFSET 3.269 0.523 -0.125
    CHANNELS 3 Yrotation Xrotation Zrotation
    JOINT LeftHandRing1
    {
        OFFSET 4.502 -0.021 -0.465
        CHANNELS 3 Yrotation Xrotation Zrotation
        JOINT LeftHandRing2
        {
            OFFSET 3.344 0.000 0.000
            CHANNELS 3 Yrotation Xrotation Zrotation
            JOINT LeftHandRing3
            {
```

```
                                        OFFSET 2.320 0.000 0.000
                                        CHANNELS 3 Yrotation Xrotation Zrotation
                                        End Site
                                        {
                                            OFFSET 1.804 0.000 0.000
                                        }
                                    }
                                }
                            }
                            JOINT LeftInHandPinky
                            {
                                OFFSET 3.071 0.456 -1.167
                                CHANNELS 3 Yrotation Xrotation Zrotation
                                JOINT LeftHandPinky1
                                {
                                    OFFSET 4.023 -0.021 -1.059
                                    CHANNELS 3 Yrotation Xrotation Zrotation
                                    JOINT LeftHandPinky2
                                    {
                                        OFFSET 2.678 0.000 0.000
                                        CHANNELS 3 Yrotation Xrotation Zrotation
                                        JOINT LeftHandPinky3
                                        {
                                            OFFSET 1.692 0.000 0.000
                                            CHANNELS 3 Yrotation Xrotation Zrotation
                                            End Site
                                            {
                                                OFFSET 1.598 0.000 0.000
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
MOTION
Frames: 4585
```

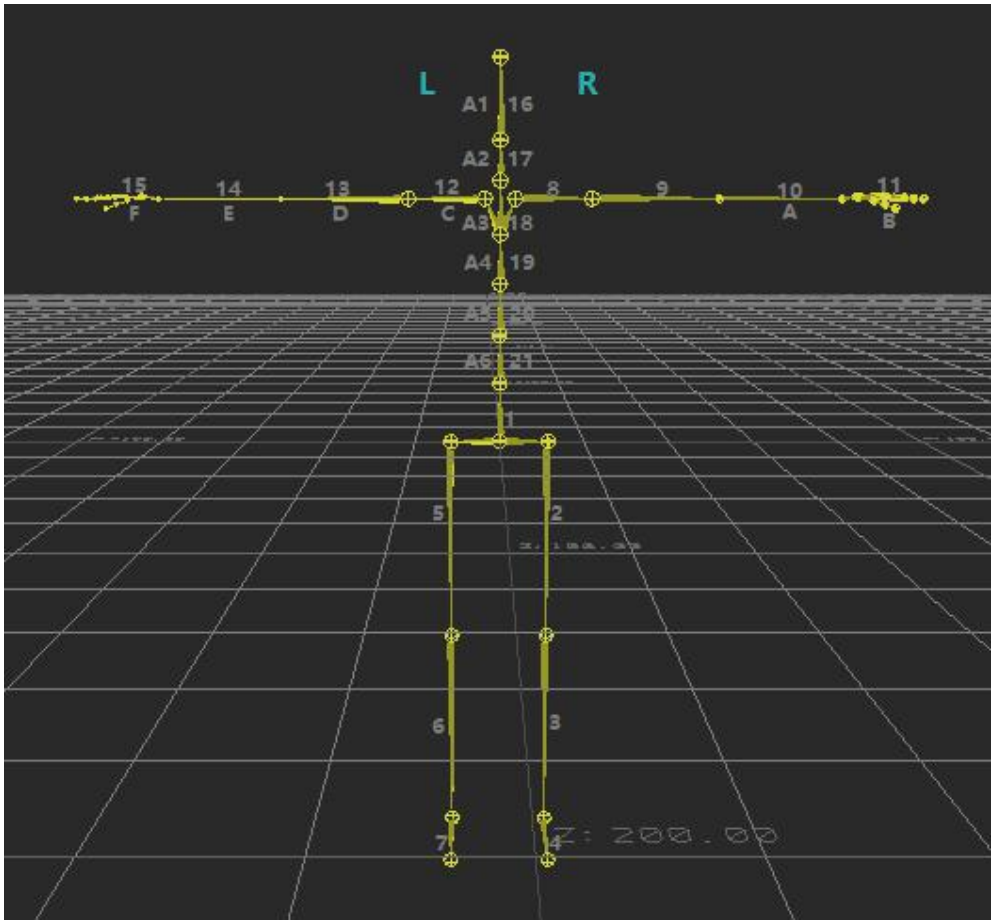
Frame Time: 0.010000

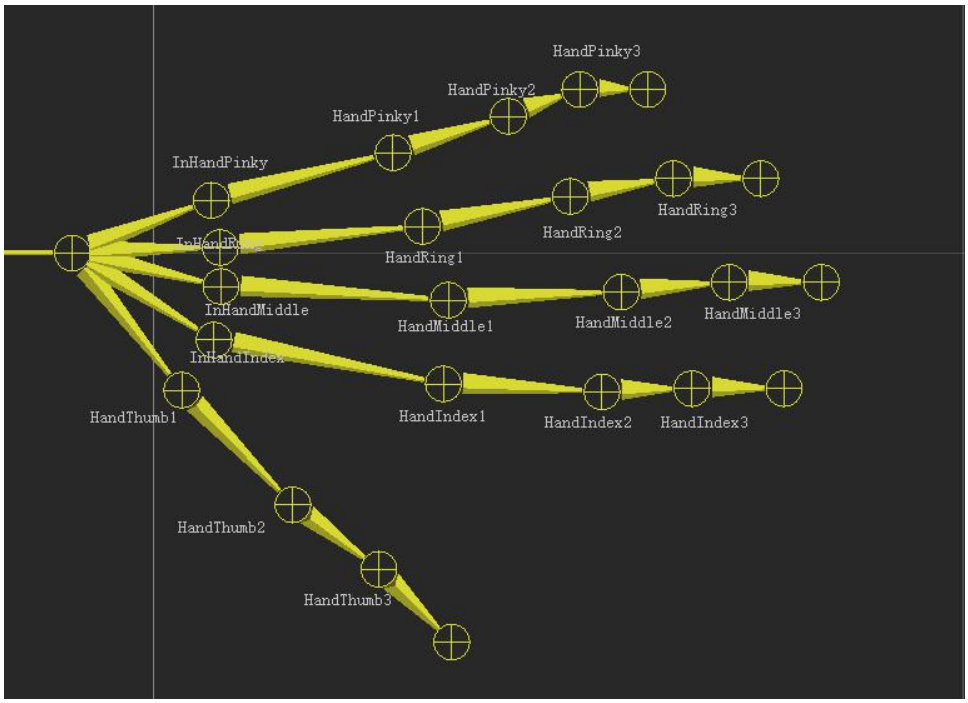
Appendix C: Bone Sequence Table

0. Hips
1. RightUpLeg
2. RightLeg
3. RightFoot
4. LeftUpLeg
5. LeftLeg
6. LeftFoot
7. RightShoulder
8. RightArm
9. RightForeArm
10. RightHand
11. LeftShoulder
12. LeftArm
13. LeftForeArm
14. LeftHand
15. Head
16. Neck1
17. Neck
18. Spine2
19. Spine1
20. Spine
21. RightHandThumb1
22. RightHandThumb2
23. RightHandThumb3
24. RightInHandIndex
25. RightHandIndex1
26. RightHandIndex2
27. RightHandIndex3
28. RightInHandMiddle
29. RightHandMiddle1
30. RightHandMiddle2
31. RightHandMiddle3
32. RightInHandRing
33. RightHandRing1
34. RightHandRing2
35. RightHandRing3

36. RightInHandPinky
37. RightHandPinky1
38. RightHandPinky2
39. RightHandPinky3
40. 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

Appendix D: 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.
D13	Yufeng Tang	10/10/2015	**Add:** Details of skeleton data and data acquisition frequency description.

			C++ demo.
D14	Yufeng Tang	23/10/2015	**Add:** Details of calculation data and data acquisition frequency description. Appendix D: Bone index for calculation data.
D15	Yuanhui He	12/11/2015	**Add:** Appendix C: Bone Sequence Table
D16	Yufeng Tang	30/12/2015	**Add:** Chinese translation. **Modify:** Appendix A: Skeleton Data Sequence in Array. **Delete:** 2 command communication API in Section 2.3.
D17	Haoyang Zhang	13/1/2017	**Add:** Data pipe transmission. Cmd pipe transmission.
	Yuanhui He	18/1/2017	Document correction
	Yuanhui He	8/2/2017	Change command header, add a reserved field to make this struct align to 8 bytes.
	Yuanhui He	11/4/2017	**Modify：** The description of the data frequency and the packets lost
	Xudong Wang	26/6/2018	**Modify:** Appendix A: Skeleton Data Sequence in Array. Appendix B: BVH Header Template Appendix C: Bone Sequence Table
D18	Xudong Wang	16/8/2018	**Add:** Add android's lib and dll