Neuron Data Reader API 指导手册

By 何元会
王旭东

16.08.2018

NeuronDataReader b18

北京诺亦腾科技有限公司

本文档为开发人员阐述了如何使用 NeuronDataReader 库以及如何应用通过该库获得的骨骼数据。

contents

Neuron Data Reader API 指导手册 1
1. 简介 4
1.1. NeuronDataReader 框架 4
1.2. 骨骼数据格式 4
1.2.1.BVH 数据 4
1.2.2.中间数据 6
1.3. 数据频率 6
1.4. 注意事项 7
2. 使用指导 8
2.1. 数据类型定义 8
2.1.1. Socket 连接状态 8
2.1.2. 数据流的版本 8
2.1.3. BVH 数据流的头 8
2.1.4. Calc Data 数据流头 9
2.1.5. 命令 ID（CmdId） 9
2.1.6. 命令定义 9
2.1.7. 命令头定义 10
2.1.8. 骨骼尺寸 10
2.1.9. BVH 旋转顺序 10
2.2. 回调函数和注册回调 11
2.2.1. 回调 BVH 数据 11
2.2.2. 回调中间数据 11
2.2.3. Socket 状态回调 11
2.3. API 使用说明 13
2.3.1. BRRegisterFrameDataCallback 13
2.3.2. BRRegisterCalculationDataCallback 13
2.3.3. BRRegisterSocketStatusCallback 13
2.3.4. BRConnectTo 14
2.3.5. BRStartUDPServiceAt 14

2.3.6. BRCloseSocket....14
2.3.7. BRGetSocketStatus....14
2.3.8. BRGetLastErrorMessage....14
2.3.9. BRConnectCmd....15
2.3.10. ZeroOut....15
2.3.11. ZeroOutAll....15
2.3.12. GetBvhRotation....15
2.3.13. BRGetBoneSize....16
2.3.14. GetBvhHeader....16
3. 问题反馈....16
Appendix A: 骨骼数据序列列表....17
Appendix B: BVH 数据头模板....19
Appendix C: Bone 序列表....28
Appendix D: 骨骼参考图....30
Revision history....31

1. 简介

北京诺亦腾科技有限公司研发的 Axis Neuron 软件可以通过 TCP/IP 或者 UDP 协议输出 BVH 动作数据和中间数据，并通过一些简单的指令获取、同步参数。 NeuronDataReader SDK 则是为简化大家从 Axis 获取数据而提供的编程接口。

1.1. NeuronDataReader 框架

NeuronDataReader 库的结构如下所示。

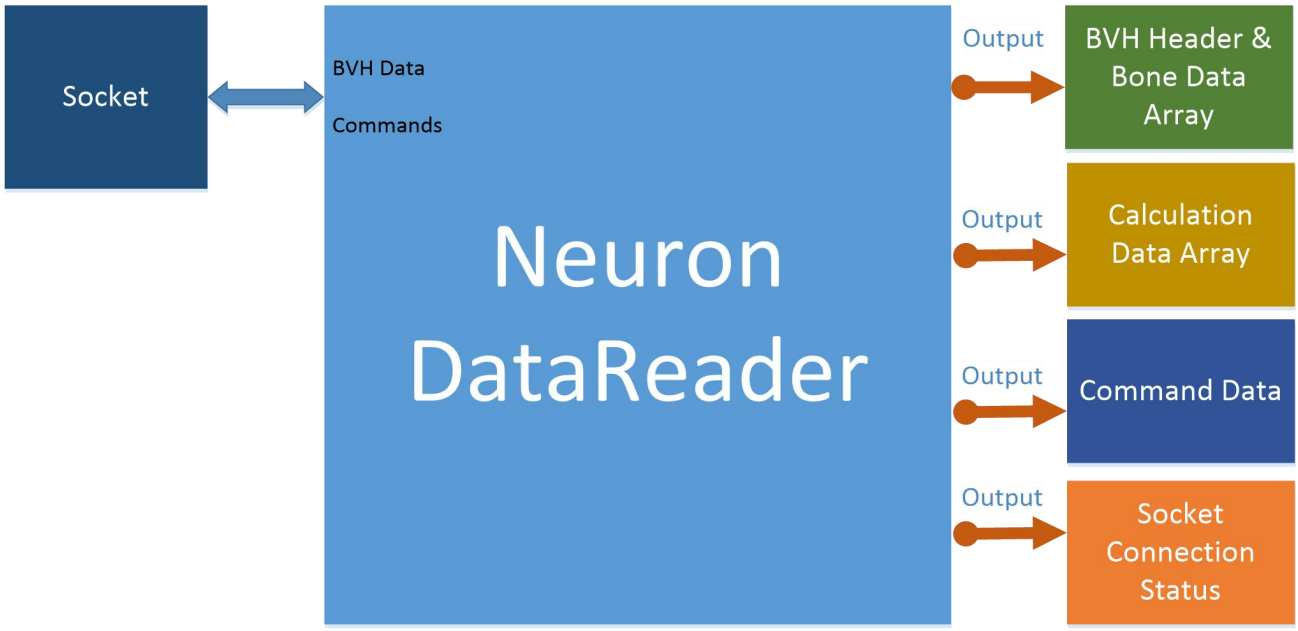


图. 1-1 NeuronDataReader 结构综述

为了兼容多种语言调用，NeuronDataReader SDK 为纯 C 函数接口。

1.2. 骨骼数据格式

在此版本中，骨骼数据包括了 BVH 数据和中间数据、参数同步指令，均以回调的方式输出。

1.2.1.BVH 数据

NeuronDataReader 以回调方式将 BVH 动作数据传递到客户端代码。参数包含骨骼数据信息头及骨骼数据，在 float 数据阵列中骨骼数据的序列如 Appendix A 所示。Appendix B 展示了 BVH 数据头的样例，在实时数据流以供参考。

NeuronDataReader 通过网络从 Axis Neuron 获取 BVH 帧数据，每一帧中的 BVH 数据包含了 59 根骨骼的全部的动作数据。

对于带有位移（displacement）的 BVH 数据，每一根骨骼包含了 6 个 float 型数据：位移（X Y Z）和旋转（默认的旋转顺序为 Y X Z）。

对于不带位移（displacement）的 BVH 数据，只有根节点（Hip）包含位移和旋转数据。其余的每一根骨骼只包含了 3 个旋转数据（默认的旋转顺序为 Y X Z）。

因此，如果用户想获取指定骨骼的信息（位置或姿态），用户可以基于下面的公式来计算相关的编号索引。

1) 带有位移（displacement）的 BVH 数据

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) 不带位移（displacement）的 BVH 数据

除了旋转数据，只有根节点包含位移数据。

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) BVH 坐标系

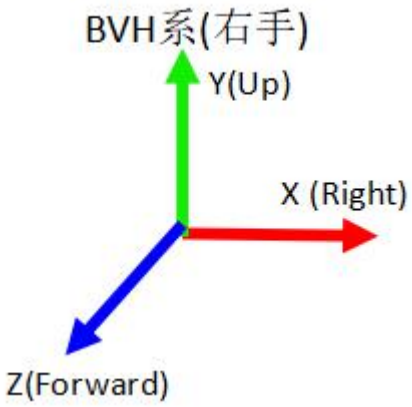


图. 1-2 BVH 坐标系

4) 带有前缀（Reference）的 BVH 数据

为了转换或旋转全部的骨骼，但同时不改变骨骼数据，设计者在 BVH 数据结构中加入了一个新的节点，作为根节点的父节点。因此，只要改变这个新节点的位移，整个骨骼模型也会随之改变；只要改变这个新节点的姿态，就可以旋转整个骨骼模型。这个新节点被称之为前缀（Reference）。

前缀（Reference）中有位移和旋转数据，共计 6 个值。

所以，对于上述的 BVH 数据，如果包含了前缀，获取骨骼数据的索引号时，需要在计算时加 6（作为偏移量）。

BVH 数据结构参见 Appendix A。

注意：每个节点的位姿数据为相对于父节点的相对数据，根节点“Hips”的位姿相对于参考点。

BVH 数据的位移量通常是常数，它体现了与 TPose 的父节点的初始位置相关的骨骼的初始位置的偏移量。因此它是一个常数。

1.2.2.中间数据

NeuronDataReader 通过网络从 Axis Neuron 获取中间数据帧，每一帧中的中间数据包含了 59 根骨骼的全部的传感器数据和动作数据以及双脚的接触状态。

对于中间数据，每一根骨骼包含了 16 个 float 型数据，它们分别是：3 个位置数据（世界坐标系中的 X Y Z 轴）、3 个速度数据（世界坐标系中的 X Y Z 轴）、4 个四元数数据（世界坐标系中的 W X Y Z 轴）、3 个加速度数据（模块坐标系中的 X Y Z 轴）和陀螺仪的 3 个数据（模块坐标系中的 X Y Z 轴）。

中间数据格式如下所示：

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

对于 b15 及之前的版本，每一帧的输出序列是从 No.1 到 No.21 的 21 根骨骼数据，紧接着的是左脚和右脚的两个接触标志位（1 表示接触地面、0 表示未接触地面，4 字节 float 类型）。自 b15 之后，中间数据扩展到 59 根骨骼，增加了左右手的手指骨骼的数据，一共 38 根，紧接在两个接触标记位之后。

骨骼列表和中间数据的输出序列参见 Appendix C 和 Appendix D。

1.3. 数据频率

从 NeuronDataReader 输出的数据的频率取决于当前用户穿戴的传感器的数量。我们规定，当所穿戴的设备节点小于 18 时，对应的采集频率为 120Hz；当所穿戴的设备节点大于等于 18 时，对应的采集频率为 60Hz。相应地，调用回调函数的频率与数据采集频率一致。

需要注意的是，在通过网络进行数据传输的过程中，存在丢帧的较小可能。即使频率固定，通过 NeuronDataReader 获取到的数据频率也可能有变化。

1.4. 注意事项

NeuronDataReader 使用回调的方式来输出数据。因此，在连接到服务器之前，必须注册回调函数用来接收相应的数据。由于用于注册的函数通常为静态函数或全局函数，当注册回调函数时，可以传入一个自定义对象（第一个参数），以便函数被回调时可以获取注册该回调函数的对象。

NeuronDataReader 中的数据处理的线程从 UI 分离出的工作线程。因此用户注册的数据接收函数不能直接访问 UI 元素。但是回调函数的数据或者状态可以被保存至一个本地数组或者缓存，因此 UI 线程可以在其他任何地方访问本地缓存的数据。

NeuronDataReader 库中有一些用来与服务器同步参数或数据的命令。

由于 C#或者 Unity 无法直接调用 C++动态的库和 API，NeuronDataReader 采用了纯 C 的接口。

2. 使用指导

NeuronDataReader 库的数据类型、句柄和程序接口如下所列。

2.1. 数据类型定义

2.1.1. Socket 连接状态

socket 连接状态的枚举类型如下所示：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. 数据流的版本

对于不同版本的 NeuronDataReader，用于通讯的数据结构在定义和结构上会有改变。数据版本用来与旧版本的 NeuronDataReader 生成的数据相匹配。

```
// 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. BVH 数据流的头

```
// 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;
```

详细说明请参考 1.2.1.

2.1.4.Calc Data 数据流头

```
// 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;
```

详细说明请参考 1.2.2.

2.1.5.命令 ID（CmdId）

```
// 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.命令定义

```
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.命令头定义

```
// 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.骨骼尺寸

```
// 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 旋转顺序

```
// 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. 回调函数和注册回调

通过回调函数，NeuronDataReader 库输出骨骼数据或者 socket 状态。因此，当接收这些数据时，应该先注册 NeuronDataReader 库中相关的回调句柄。

2.2.1.回调 BVH 数据

```
    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.回调中间数据

```
    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 状态回调

```
    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.
```

注意：由于 NeuronDataReader 中的数据处理是多线程异步的，数据接收回调函数不能直接访问 UI 元素。如果需要将数据用于 UI 线程，建议将从回调函数得到的数据保存至一个本地的数组。

2.3. API 使用说明

2.3.1.BRRegisterFrameDataCallback

注册 BVH 数据接收回调句柄。

```
// Register data-receiving callback handle.
NEURONDATAREADER_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

注册 calculation 数据回调句柄。

```
// Register data-receiving callback handle.
NEURONDATAREADER_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

注册 socket 状态回调句柄。

```
// Register socket status callback
NEURONDATAREADER_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

由给定的 IP 地址和端口连接至服务器。

```
// Connect to server
NEURONDATAREADER_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

由于 Axis Neuron 可以通过 TCP/IP 或者 UDP 输出数据，NeuronDataReader 也可以读取和解析两种 socket 数据类型。函数 BRStartUDPServiceAt 用来开始一个监听和接收由服务器发出的数据的服务。

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

2.3.6.BRCloseSocket

停止数据接收服务。值得注意的是，需要在程序退出前调用这个函数来断开/停止服务器的服务，否则程序将由于数据接收线程的阻塞而不能退出。

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

2.3.7.BRGetSocketStatus

检查 socket 状态。实际上通过 socket 回调句柄，这个函数有相同的输出状态。如果已经注册 socket 状态回调句柄，这个函数就不需要使用了。

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

Return Values

Return the status of referred socket.

2.3.8.BRGetLastErrorMessage

一旦出现错误，通过调用'BRGetLastErrorMessage'函数来获取错误信息。

```
NEURONDATAREADER_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

由给定的 IP 地址和端口连接至服务器的数据通道。

```
// 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

通过已经建立的命令通道向 Axis 发送命令。

```
// 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

通过已经建立的命令通道向 Axis 发送命令。

```
// 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

通过已经建立的命令通道向 Axis 发送命令。

```
// 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

通过已经建立的命令通道向 Axis 发送命令。

```
// 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

通过已经建立的命令通道向 Axis 发送命令。

```
// 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. 问题反馈

如果您在使用中遇到了任何本文档中未提及的 bug 或者问题，请发邮件至 xudong.wang@noitom.com 或者 yuanhui.he@noitom.com

谢谢！

Appendix A: 骨骼数据序列列表

	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 数据头模板

```
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
```

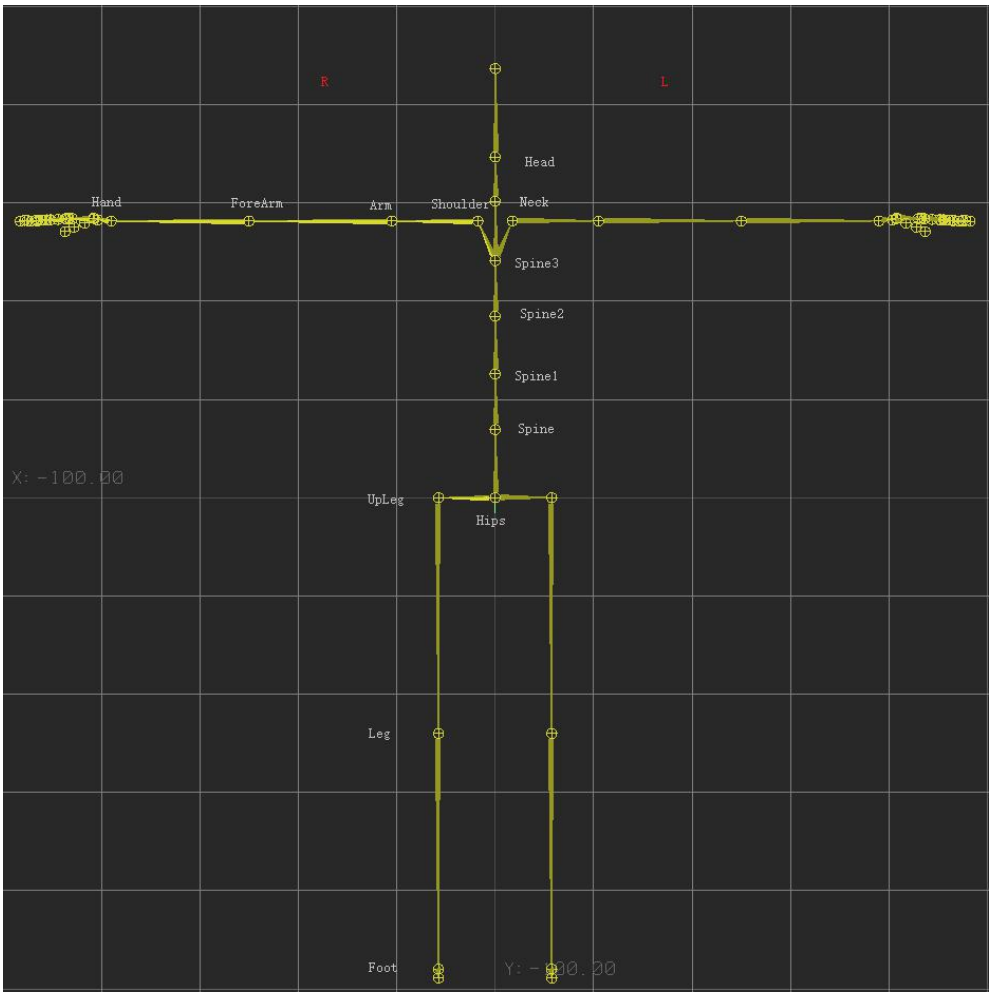
Frame Time: 0.010000

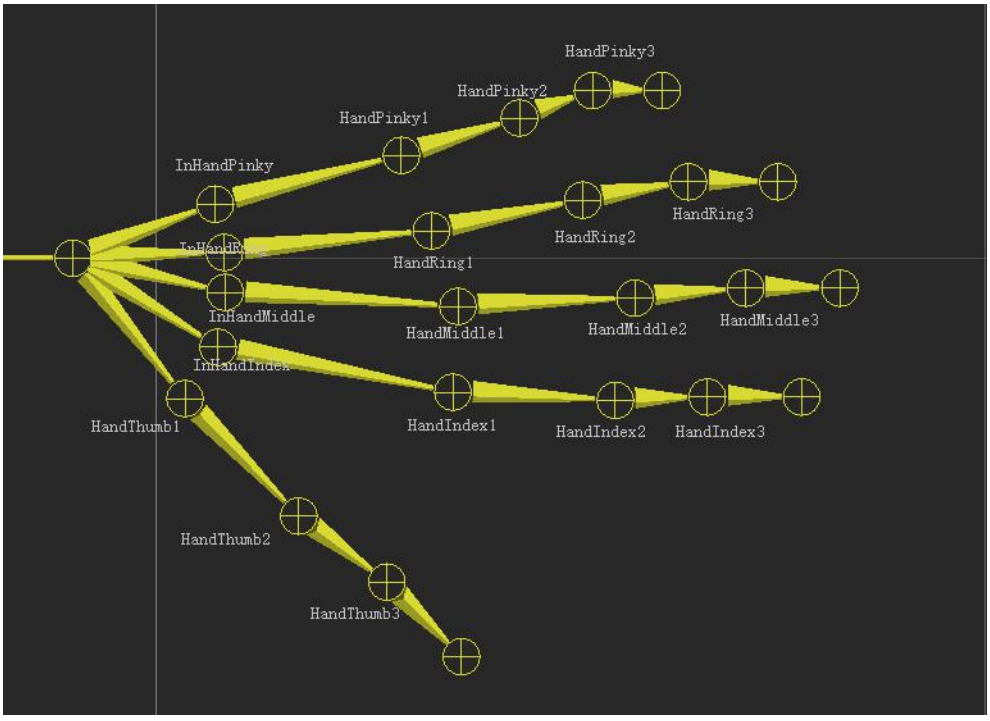
Appendix C: Bone 序列表

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: 骨骼参考图





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.
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