|  |
| --- |
| Noitom Technology Co., Ltd. |
| PNLib Runtime API Documentation |

|  |
| --- |
| Yuanhui  2014/12/3 |

Contents

[1 Overview 9](#_Toc418584003)

[1.1 PNLib SDK framework 9](#_Toc418584004)

[1.2 Flowchart of calling PNLib SDK 10](#_Toc418584005)

[1.3 Other things must know 11](#_Toc418584006)

[1.3.1 Sensor 11](#_Toc418584007)

[1.3.2 Coordinates 12](#_Toc418584008)

[1.3.3 Skeleton system 13](#_Toc418584009)

[1.3.4 Node modes 3](#_Toc418584010)

[1.3.5 Diagram of wear 3](#_Toc418584011)

[2 Reference 5](#_Toc418584012)

[2.1 Data type definitions 5](#_Toc418584013)

[2.1.1 SensorSuitTypes 5](#_Toc418584014)

[2.1.2 SensorCombinationModes 5](#_Toc418584015)

[2.1.3 RunningMode 6](#_Toc418584016)

[2.1.4 SensorAcceleratorTypes 6](#_Toc418584017)

[2.1.5 RotateOrders 6](#_Toc418584018)

[2.1.6 OutputQuaternionTypes 7](#_Toc418584019)

[2.1.7 OutputAccelerationTypes 7](#_Toc418584020)

[2.1.8 OutputGyroType 8](#_Toc418584021)

[2.1.9 BvhDataStreamTypes 8](#_Toc418584022)

[2.1.10 CalculatedDataStreamTypes 9](#_Toc418584023)

[2.1.11 ConstraintPoint 9](#_Toc418584024)

[2.1.12 MagneticImmuneLevel 10](#_Toc418584025)

[2.1.13 CalibrationTypes 10](#_Toc418584026)

[2.1.14 PNLibVersion 11](#_Toc418584027)

[2.1.15 OutputDataVersion 11](#_Toc418584028)

[2.1.16 BvhOutputBinaryHeader 12](#_Toc418584029)

[2.1.17 CalculationDataHeader 13](#_Toc418584030)

[2.1.18 Quaternion4\_t 13](#_Toc418584031)

[2.1.19 Vector3\_t 14](#_Toc418584032)

[2.1.20 BoneMap 14](#_Toc418584033)

[2.1.21 BoneDimension 15](#_Toc418584034)

[2.1.22 ContactStatus 16](#_Toc418584035)

[2.1.23 FrameContactData 16](#_Toc418584036)

[2.1.24 RawFileTime 16](#_Toc418584037)

[2.1.25 RawFileAvatarInfo 17](#_Toc418584038)

[2.1.26 RawFileInfo 18](#_Toc418584039)

[2.1.27 CalibrationData 18](#_Toc418584040)

[2.1.28 SmoothFactors 19](#_Toc418584041)

[2.2 Callbacks 20](#_Toc418584042)

[2.2.1 PNEventCalibrationProgressCallback 20](#_Toc418584043)

[2.2.2 PNEventPlayProgressCallback 20](#_Toc418584044)

[2.2.3 PNEventContactNotificationCallback 20](#_Toc418584045)

[2.2.4 PNEventRawDataParsedCallback 21](#_Toc418584046)

[2.2.5 PNEventCalculatedStringDataCallback 21](#_Toc418584047)

[2.2.6 PNEventCalculatedBinaryDataCallback 22](#_Toc418584048)

[2.2.7 PNEventBVHStringDataBoardcastCallback 22](#_Toc418584049)

[2.2.8 PNEventBVHBinaryDataBoardcastCallback 22](#_Toc418584050)

[2.2.9 PNEventBVHMatrixDataBoardcastCallback 23](#_Toc418584051)

[2.2.10 PNEventConstraintDataCallback 23](#_Toc418584052)

[2.2.11 PNEventActionRecognizationDataStringStreamCallback 24](#_Toc418584053)

[2.2.12 PNEventBodyMassVectorStringCallback 24](#_Toc418584054)

[2.2.13 PNEventBodySwingVectorStringCallback 25](#_Toc418584055)

[2.3 API reference 25](#_Toc418584056)

[2.3.1 PNGetLibVersion 25](#_Toc418584057)

[2.3.2 PNLibInit 25](#_Toc418584058)

[2.3.3 PNGetLastErrorCode 25](#_Toc418584059)

[2.3.4 PNGetLastErrorMessage 26](#_Toc418584060)

[2.3.5 PNSetSensorSuitType 26](#_Toc418584061)

[2.3.6 PNGetStandardBoneTable 26](#_Toc418584062)

[2.3.7 PNSetSensorCombinationMode 26](#_Toc418584063)

[2.3.8 PNSetDataFolders 27](#_Toc418584064)

[2.3.9 PNSetDataAcquisitionFrequency 27](#_Toc418584065)

[2.3.10 PNGetDataAcquisitionFrequency 28](#_Toc418584066)

[2.3.11 PNSetRunningMode 28](#_Toc418584067)

[2.3.12 PNGetRunningMode 28](#_Toc418584068)

[2.3.13 PNLoadCalibrationData 28](#_Toc418584069)

[2.3.14 PNEnableClimbContact 29](#_Toc418584070)

[2.3.15 PNResetClimbContact 29](#_Toc418584071)

[2.3.16 PNEnableMagneticImmune 29](#_Toc418584072)

[2.3.17 PNSetSpineSmoothFactors 30](#_Toc418584073)

[2.3.18 PNSetSensorAcceleratorType 30](#_Toc418584074)

[2.3.19 PNGetSensorAcceleratorType 30](#_Toc418584075)

[2.3.20 PNReleaseScene 31](#_Toc418584076)

[2.3.21 PNRegisterCalibrationProgressHandle 31](#_Toc418584077)

[2.3.22 PNRegisterBvhStringDataBoardcastHandle 32](#_Toc418584078)

[2.3.23 PNRegisterBvhBinaryDataBoardcastHandle 32](#_Toc418584079)

[2.3.24 PNRegisterBvhMatrixDataBoardcastHandle 33](#_Toc418584080)

[2.3.25 PNSetBvhDataBlockBoardcastType 34](#_Toc418584081)

[2.3.26 PNEnableBvhDataBoardcast 35](#_Toc418584082)

[2.3.27 PNSetCalculatedDataBlockBoardcastType 35](#_Toc418584083)

[2.3.28 PNRegisterCalculatedStringDataBoardcastHandle 35](#_Toc418584084)

[2.3.29 PNRegisterCalculatedBinaryDataBoardcastHandle 36](#_Toc418584085)

[2.3.30 PNSetCalculatedQuaternionDataType 36](#_Toc418584086)

[2.3.31 PNSetCalculatedAccelerationDataType 36](#_Toc418584087)

[2.3.32 PNSetCalculatedGyroDataType 36](#_Toc418584088)

[2.3.33 PNEnableCalculationDataBoardcast 37](#_Toc418584089)

[2.3.34 PNRegisterRawDataPlayingProgressHandle 37](#_Toc418584090)

[2.3.35 PNRegisterPlayingRawDataParsedHandle 38](#_Toc418584091)

[2.3.36 PNRegisterContactNotification 38](#_Toc418584092)

[2.3.37 PNRegisterContactEditCallback 38](#_Toc418584093)

[2.3.38 PNEnableContactNotification 39](#_Toc418584094)

[2.3.39 PNRegisterActionRecognizationStringDataBoardcastHandle 39](#_Toc418584095)

[2.3.40 PNRegisterBodyMassVectorStringCallback 39](#_Toc418584096)

[2.3.41 PNRegisterBodySwingVectorStringCallback 40](#_Toc418584097)

[2.3.42 PNCreateAvatar 40](#_Toc418584098)

[2.3.43 PNRemoveAvatar 40](#_Toc418584099)

[2.3.44 PNGetAvatarCount 41](#_Toc418584100)

[2.3.45 PNSetAvatarName 41](#_Toc418584101)

[2.3.46 PNGetAvatarName 41](#_Toc418584102)

[2.3.47 PNSetBoneDimensions 41](#_Toc418584103)

[2.3.48 PNGetBoneDimensions 42](#_Toc418584104)

[2.3.49 PNGetBoneLength 42](#_Toc418584105)

[2.3.50 PNBindSensor 43](#_Toc418584106)

[2.3.51 PNRemoveSensor 43](#_Toc418584107)

[2.3.52 PNIsBindingSensor 44](#_Toc418584108)

[2.3.53 PNCheckSensorBindingMode 44](#_Toc418584109)

[2.3.54 PNResetBoneMapping 45](#_Toc418584110)

[2.3.55 PNGetBoneName 45](#_Toc418584111)

[2.3.56 PNGetBoneNameBySensorId 45](#_Toc418584112)

[2.3.57 PNGetSensorId 46](#_Toc418584113)

[2.3.58 PNGetBoneIdBySensor 46](#_Toc418584114)

[2.3.59 PNGetHipWidth 46](#_Toc418584115)

[2.3.60 PNGetHipHight 47](#_Toc418584116)

[2.3.61 PNGetShoulderWidth 47](#_Toc418584117)

[2.3.62 PNGetHeelHight 47](#_Toc418584118)

[2.3.63 PNGetInitiationDirection 48](#_Toc418584119)

[2.3.64 PNGetInitiationLeftDirection 48](#_Toc418584120)

[2.3.65 PNCanCalibratePose 48](#_Toc418584121)

[2.3.66 PNGetCalibrationData 49](#_Toc418584122)

[2.3.67 PNSetCalibrationData 49](#_Toc418584123)

[2.3.68 PNClearIntegralState 49](#_Toc418584124)

[2.3.69 PNSetDataOutputFrequencyRatio 49](#_Toc418584125)

[2.3.70 PNGetDataOutputFrequencyRatio 50](#_Toc418584126)

[2.3.71 PNGetKalmanParams 50](#_Toc418584127)

[2.3.72 PNSetKalmanParams 50](#_Toc418584128)

[2.3.73 PNResetKalmanParams 51](#_Toc418584129)

[2.3.74 PNSetPDamping 51](#_Toc418584130)

[2.3.75 PNGetPDamping 51](#_Toc418584131)

[2.3.76 PNSetJointStiffness 51](#_Toc418584132)

[2.3.77 PNSetStepStiffness 51](#_Toc418584133)

[2.3.78 PNSetStepConstraint 52](#_Toc418584134)

[2.3.79 PNGetStiffnessPercent 52](#_Toc418584135)

[2.3.80 PNEnableSmoothFilter 52](#_Toc418584136)

[2.3.81 PNIsSmoothFilterEnabled 53](#_Toc418584137)

[2.3.82 PNSetSmoothFactor 53](#_Toc418584138)

[2.3.83 PNGetSmoothFactor 53](#_Toc418584139)

[2.3.84 PNCreateBvhPlayer 54](#_Toc418584140)

[2.3.85 PNSetBvhPlayerCameras 54](#_Toc418584141)

[2.3.86 PNEnableBvhPlayerCameraBind 54](#_Toc418584142)

[2.3.87 PNBvhPlayerResizeToParent 54](#_Toc418584143)

[2.3.88 PNCloseBvhPlayer 55](#_Toc418584144)

[2.3.89 PNEnableMassShowing 55](#_Toc418584145)

[2.3.90 PNEnableRendering 55](#_Toc418584146)

[2.3.91 PNPushData 55](#_Toc418584147)

[2.3.92 PNPushDataForAvatar 56](#_Toc418584148)

[2.3.93 PNEnableLostDataFitting 56](#_Toc418584149)

[2.3.94 PNClearCalibrationBufferedData 56](#_Toc418584150)

[2.3.95 PNCalibrateAvatar 56](#_Toc418584151)

[2.3.96 PNCalibrateAllAvatars 57](#_Toc418584152)

[2.3.97 PNClearAllAvatarCalibrationBufferedData 57](#_Toc418584153)

[2.3.98 PNGetSensorReceivingStatus 57](#_Toc418584154)

[2.3.99 PNGetRawFileInfo 57](#_Toc418584155)

[2.3.100 PNOpenRawDataFile 58](#_Toc418584156)

[2.3.101 PNRawDataPlayGetTotalFrames 58](#_Toc418584157)

[2.3.102 PNGetSensorSuitType 58](#_Toc418584158)

[2.3.103 PNGetSensorCombinationMode 58](#_Toc418584159)

[2.3.104 PNRawDataPlayGetCurrentPlayingPosition 59](#_Toc418584160)

[2.3.105 PNRawDataPlaySetPlayingPosition 59](#_Toc418584161)

[2.3.106 PNRawDataPlaySetSpeed 59](#_Toc418584162)

[2.3.107 PNRawDataPlayStart 59](#_Toc418584163)

[2.3.108 PNRawDataPlayPause 59](#_Toc418584164)

[2.3.109 PNRawDataPlayStop 59](#_Toc418584165)

[2.3.110 PNRawDataPlayEnableReversePlaying 59](#_Toc418584166)

[2.3.111 PNRawDataPlaySetToPrev 60](#_Toc418584167)

[2.3.112 PNRawDataPlaySetToNext 60](#_Toc418584168)

[2.3.113 PNCloseRawDataFile 60](#_Toc418584169)

[2.3.114 PNEditContact 60](#_Toc418584170)

[2.3.115 PNGetContactStatus 61](#_Toc418584171)

[2.3.116 PNBatchEditContact 61](#_Toc418584172)

[2.3.117 PNBatchResetContactEditStatus 62](#_Toc418584173)

[2.3.118 PNResetContactEditStatus 62](#_Toc418584174)

[2.3.119 PNFeetConstraintOptimization 62](#_Toc418584175)

[2.3.120 PNExportRawData 63](#_Toc418584176)

[2.3.121 PNStopExportRawData 63](#_Toc418584177)

[2.3.122 PNExportCalculationData 63](#_Toc418584178)

[2.3.123 PNStopExportCalculationData 63](#_Toc418584179)

[2.3.124 PNExportBvhData 64](#_Toc418584180)

[2.3.125 PNStopExportBvhData 64](#_Toc418584181)

[2.3.126 PNExportFbxData 64](#_Toc418584182)

[2.3.127 PNStopExportFbxData 64](#_Toc418584183)

[2.3.128 PNSetBvhDataFormat 65](#_Toc418584184)

[2.3.129 PNSetBvhDataWithReference 65](#_Toc418584185)

[2.3.130 PNEnableBvhDataGlobalDisplacement 65](#_Toc418584186)

[2.3.131 PNRotateFaceDirection 65](#_Toc418584187)

[2.3.132 PNRotateModel 66](#_Toc418584188)

[2.3.133 PNZeroOutAllAvatar 66](#_Toc418584189)

[2.3.134 PNZeroOutPosition 66](#_Toc418584190)

[2.3.135 PNEnableActionRecognition 66](#_Toc418584191)

# Overview

## PNLib SDK framework

PNLib SDK(PNLib Runtime API) integrate a series of functions including reading and analyzing raw data, data fusion, format conversion, motion capture data output, model driven etc. User could achieve complex features by simple configuration of function calling. The coding work is minimal.

According to the classification, the output data from PNLib SDK three levels:

* Level 1: Sensor data

For analysis and processing used a single sensor;

* Level 2: Calculation data

Analysis data for labaray user. Include position, velocity, raw pose quaternion, raw acceleration data raw gyro data of every sensor;

* Level 3: Motion data

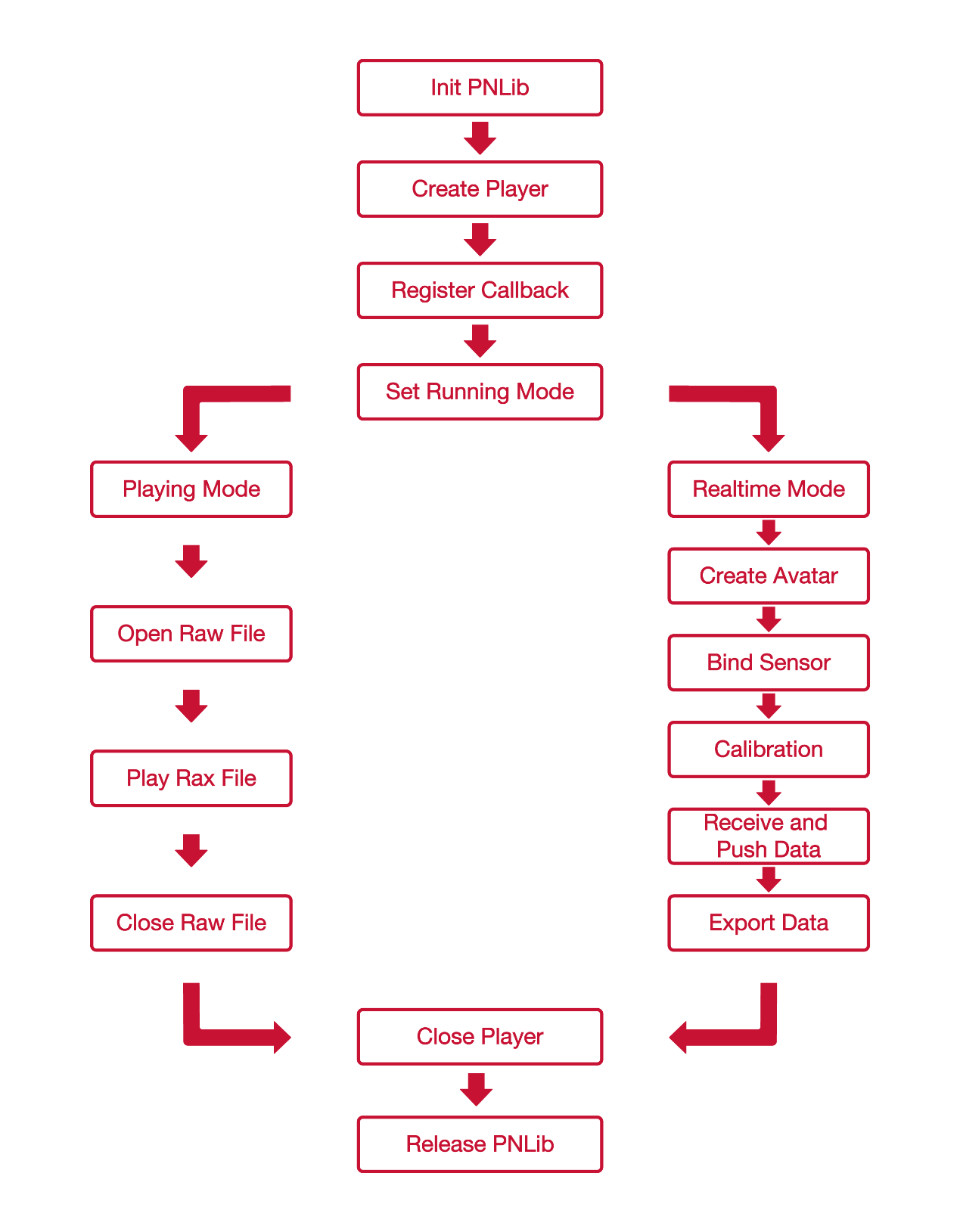
Output is mainly for action data integration of Motion Capture.



PNLib SDK supports many develop languages and interface library, such as C/C++/MFC, WPF/C#, Mac Cocoa, and supports game engines such as Unity, Unreal Engine and Unigine.

## Flowchart of calling PNLib SDK

The main steps of calling PNLib contains initing PNlLib, creating player, registering callback needed, recording motion caption data or playing raw file, closing player and releasing PNLib are as below:



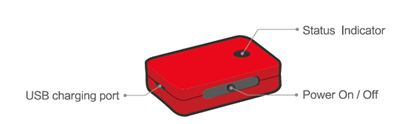
## Other things must know

### Sensor

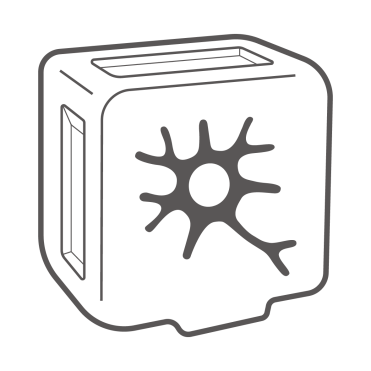
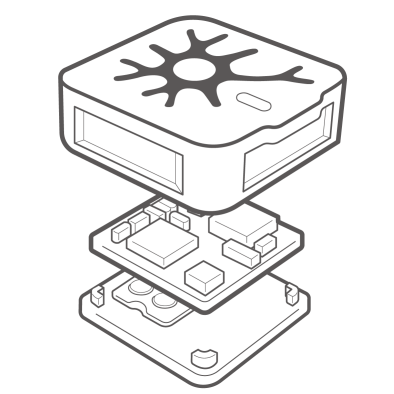
Perception Mocap System contains two types of sensor suit: Legacy and Neuron:

1. Legacy





1. Neuron

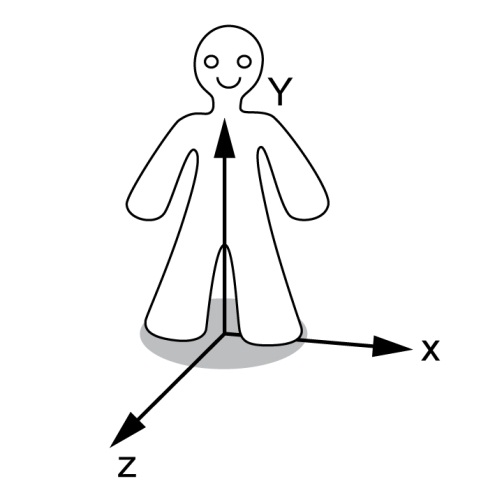
 

### Coordinates

(1)BVH data coordinate definition

Unit: centimeter

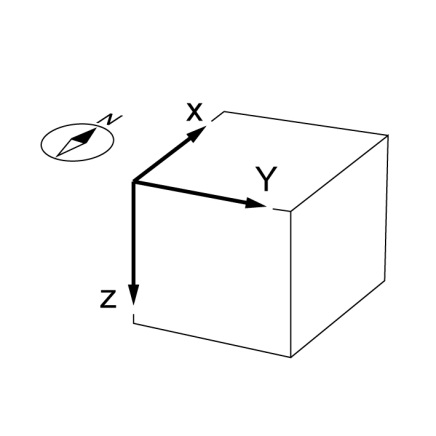
The coordinate system is as illustrated: X axis points to the left of avatar, Y axis point to the up of avatar, Z axis point to the front of avatar.



(2)Calculation data coordinate definition

Unit: meter

X axis point to the north, Y axis point to the east, Z axis point to the ground.



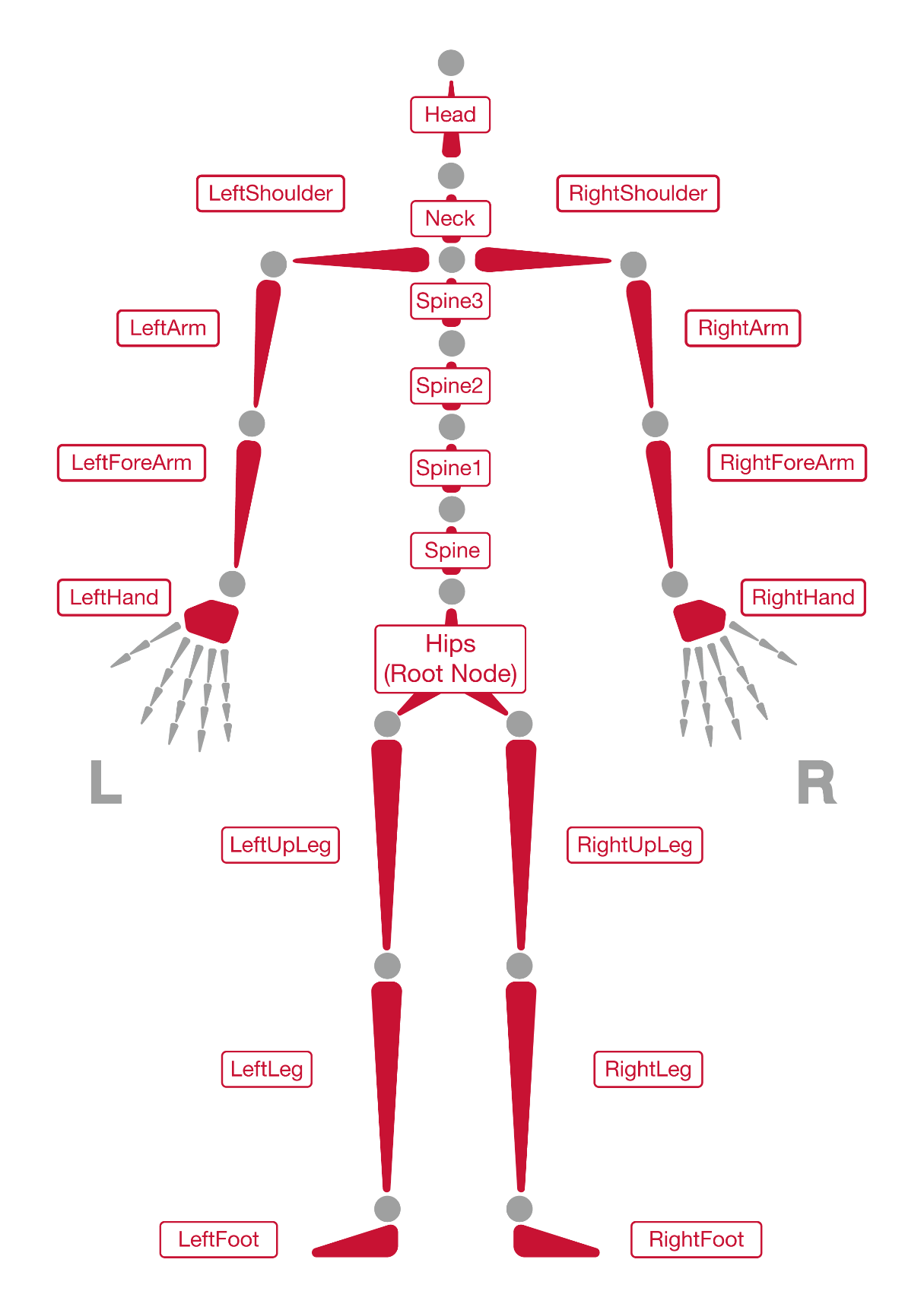
### Skeleton system

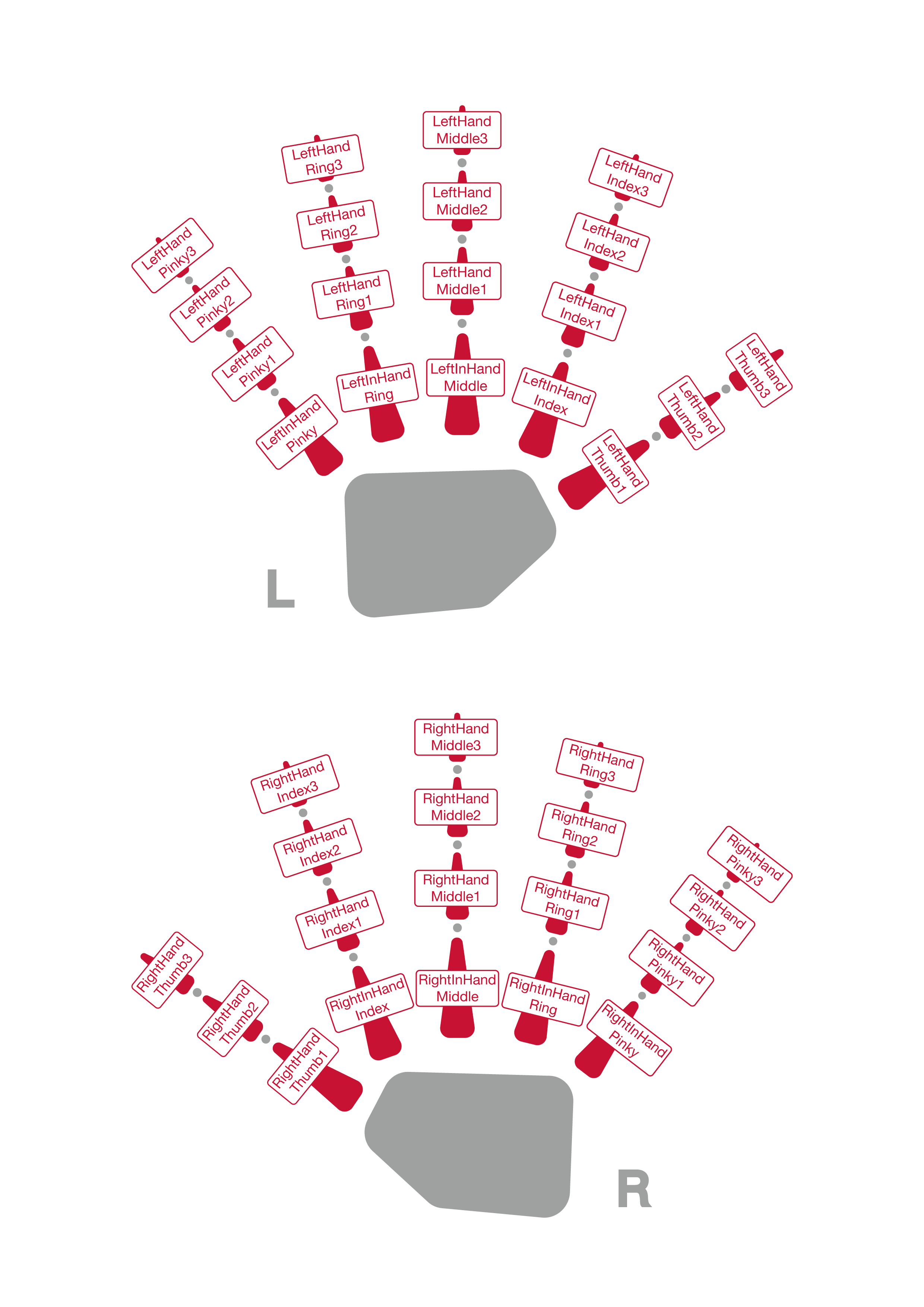
There are 167 bones in standard bone table. Legacy can accquire 21 bones’ data, and Neuron can accquire 59 bones’ data include fingers.

#### Standard Bone Table

|  |  |
| --- | --- |
| BoneIndex | BoneName |
| 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 | Neck |
| 17 | Spine3 |
| 18 | Spine2 |
| 19 | Spine1 |
| 20 | Spine |
| 21 | RightToeBase |
| 22 | LeftToeBase |
| 23 | RightFingerBase |
| 24 | RightInHandThumb |
| 25 | RightHandThumb1 |
| 26 | RightHandThumb2 |
| 27 | RightHandThumb3 |
| 28 | RightHandThumb4 |
| 29 | RightInHandIndex |
| 30 | RightHandIndex1 |
| 31 | RightHandIndex2 |
| 32 | RightHandIndex3 |
| 33 | RightHandIndex4 |
| 34 | RightInHandMiddle |
| 35 | RightHandMiddle1 |
| 36 | RightHandMiddle2 |
| 37 | RightHandMiddle3 |
| 38 | RightHandMiddle4 |
| 39 | RightInHandRing |
| 40 | RightHandRing1 |
| 41 | RightHandRing2 |
| 42 | RightHandRing3 |
| 43 | RightHandRing4 |
| 44 | RightInHandPinky |
| 45 | RightHandPinky1 |
| 46 | RightHandPinky2 |
| 47 | RightHandPinky3 |
| 48 | RightHandPinky4 |
| 49 | RightInHandExtraFinger |
| 50 | RightHandExtraFinger1 |
| 51 | RightHandExtraFinger2 |
| 52 | RightHandExtraFinger3 |
| 53 | RightHandExtraFinger4 |
| 54 | LeftFingerBase |
| 55 | LeftInHandThumb |
| 56 | LeftHandThumb1 |
| 57 | LeftHandThumb2 |
| 58 | LeftHandThumb3 |
| 59 | LeftHandThumb4 |
| 60 | LeftInHandIndex |
| 61 | LeftHandIndex1 |
| 62 | LeftHandIndex2 |
| 63 | LeftHandIndex3 |
| 64 | LeftHandIndex4 |
| 65 | LeftInHandMiddle |
| 66 | LeftHandMiddle1 |
| 67 | LeftHandMiddle2 |
| 68 | LeftHandMiddle3 |
| 69 | LeftHandMiddle4 |
| 70 | LeftInHandRing |
| 71 | LeftHandRing1 |
| 72 | LeftHandRing2 |
| 73 | LeftHandRing3 |
| 74 | LeftHandRing4 |
| 75 | LeftInHandPinky |
| 76 | LeftHandPinky1 |
| 77 | LeftHandPinky2 |
| 78 | LeftHandPinky3 |
| 79 | LeftHandPinky4 |
| 80 | LeftInHandExtraFinger |
| 81 | LeftHandExtraFinger1 |
| 82 | LeftHandExtraFinger2 |
| 83 | LeftHandExtraFinger3 |
| 84 | LeftHandExtraFinger4 |
| 85 | RightInFootThumb |
| 86 | RightFootThumb1 |
| 87 | RightFootThumb2 |
| 88 | RightFootThumb3 |
| 89 | RightFootThumb4 |
| 90 | RightInFootIndex |
| 91 | RightFootIndex1 |
| 92 | RightFootIndex2 |
| 93 | RightFootIndex3 |
| 94 | RightFootIndex4 |
| 95 | RightInFootMiddle |
| 96 | RightFootMiddle1 |
| 97 | RightFootMiddle2 |
| 98 | RightFootMiddle3 |
| 99 | RightFootMiddle4 |
| 100 | RightInFootRing |
| 101 | RightFootRing1 |
| 102 | RightFootRing2 |
| 103 | RightFootRing3 |
| 104 | RightFootRing4 |
| 105 | RightInFootPinky |
| 106 | RightFootPinky1 |
| 107 | RightFootPinky2 |
| 108 | RightFootPinky3 |
| 109 | RightFootPinky4 |
| 110 | RightInFootExtraFinger |
| 111 | RightFootExtraFinger1 |
| 112 | RightFootExtraFinger2 |
| 113 | RightFootExtraFinger3 |
| 114 | RightFootExtraFinger4 |
| 115 | LeftInFootThumb |
| 116 | LeftFootThumb1 |
| 117 | LeftFootThumb2 |
| 118 | LeftFootThumb3 |
| 119 | LeftFootThumb4 |
| 120 | LeftInFootIndex |
| 121 | LeftFootIndex1 |
| 122 | LeftFootIndex2 |
| 123 | LeftFootIndex3 |
| 124 | LeftFootIndex4 |
| 125 | LeftInFootMiddle |
| 126 | LeftFootMiddle1 |
| 127 | LeftFootMiddle2 |
| 128 | LeftFootMiddle3 |
| 129 | LeftFootMiddle4 |
| 130 | LeftInFootRing |
| 131 | LeftFootRing1 |
| 132 | LeftFootRing2 |
| 133 | LeftFootRing3 |
| 134 | LeftFootRing4 |
| 135 | LeftInFootPinky |
| 136 | LeftFootPinky1 |
| 137 | LeftFootPinky2 |
| 138 | LeftFootPinky3 |
| 139 | LeftFootPinky4 |
| 140 | LeftInFootExtraFinger |
| 141 | LeftFootExtraFinger1 |
| 142 | LeftFootExtraFinger2 |
| 143 | LeftFootExtraFinger3 |
| 144 | LeftFootExtraFinger4 |
| 145 | Neck1 |
| 146 | Neck2 |
| 147 | Neck3 |
| 148 | Neck4 |
| 149 | Neck5 |
| 150 | Neck6 |
| 151 | Neck7 |
| 152 | Neck8 |
| 153 | Neck9 |
| 154 | Spine4 |
| 155 | Spine5 |
| 156 | Spine6 |
| 157 | Spine7 |
| 158 | Spine8 |
| 159 | Spine9 |
| 160 | Props0 |
| 161 | Props1 |
| 162 | Props2 |
| 163 | Props3 |
| 164 | Props4 |
| 165 | RightShoulderExtra |
| 166 | LeftShoulderExtra |

#### Skeleton Nodes





### Node modes

There are 3 kinds of sensor combination modes:

1. SC\_ArmOnly mode:

2 nodes of left upper arm and left fore arm, or, right upper arm and right forearm are necessary.

1. SC\_UpperBody mode:

4 nodes of chest, hips, left upper arm and left forearm, or, chest, hips, right upper arm and right forearm are necessary.

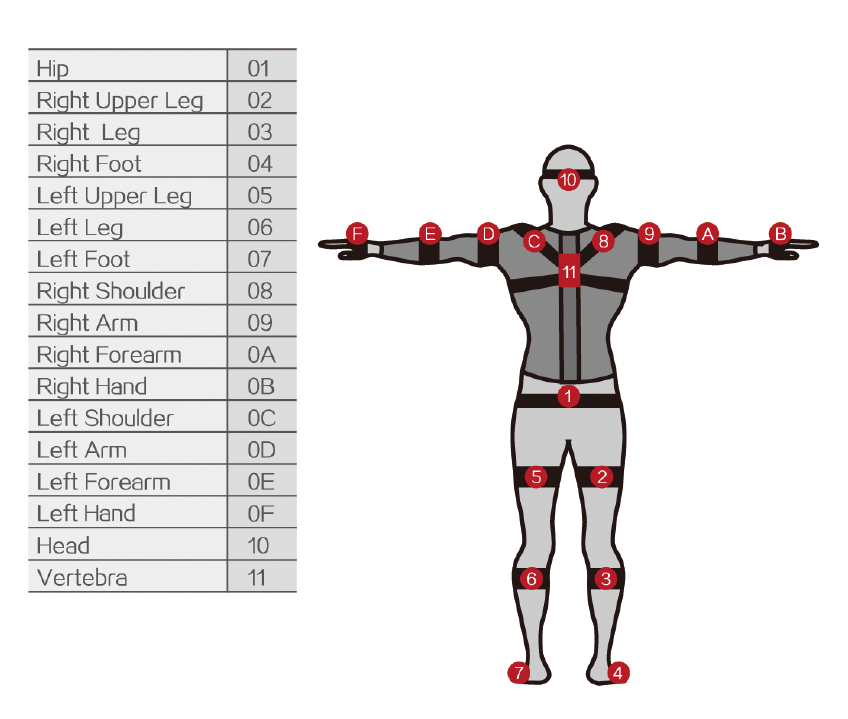
1. SC\_FullBody mode:

6 nodes of left upper leg, left leg, right upper leg, right leg, hips and chest sensor are necessary.

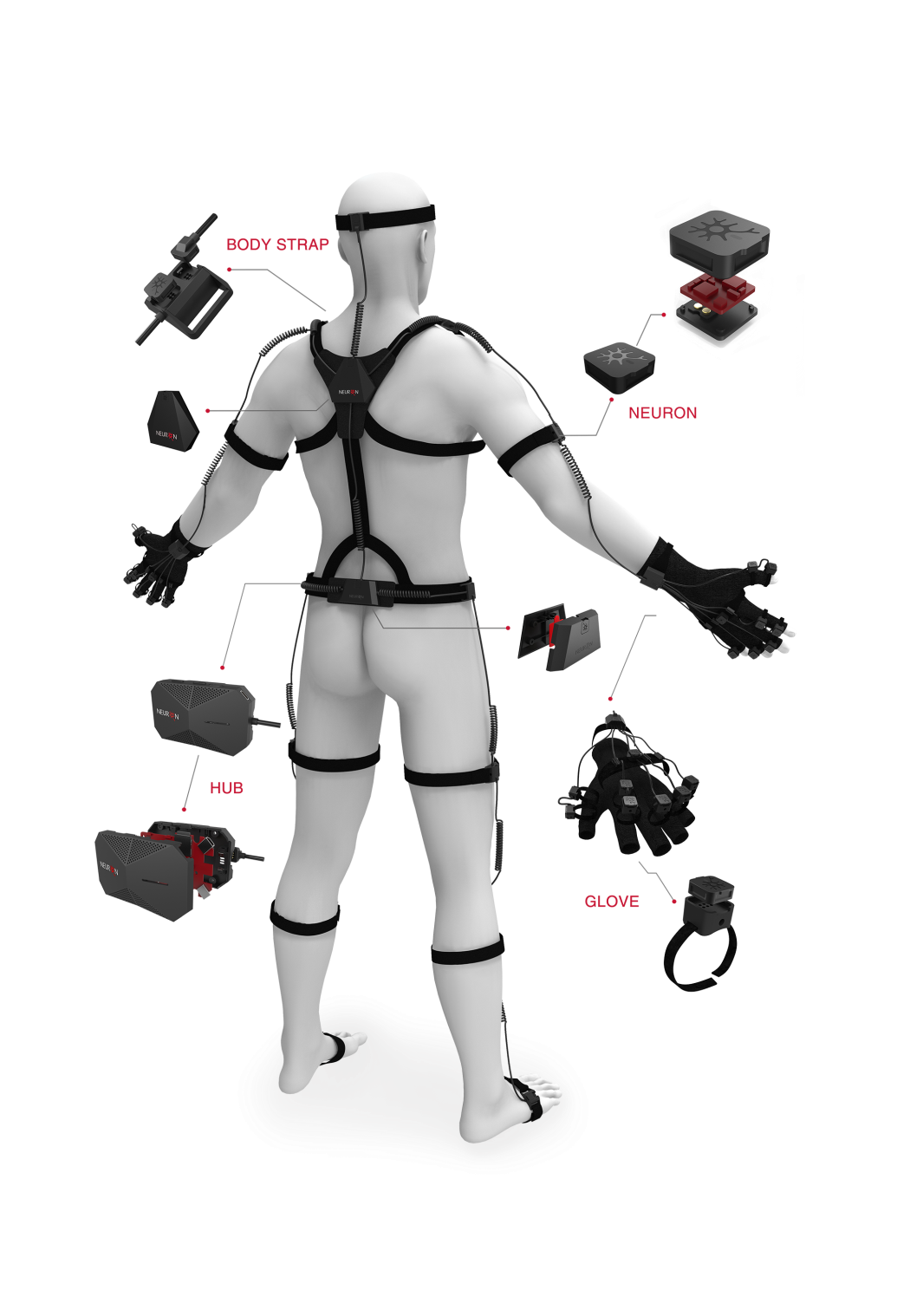
### Diagram of wear

1. Legacy



1. Neuron



# Reference

## Data type definitions

### SensorSuitTypes

Sensor suit types at realtime mode.

typedef enum \_SensorSuitTypes

{

SS\_LegacySensors,

SS\_NeuronSensors,

SS\_Unknown,

}SensorSuitTypes;

**Members**

**SS\_LegacySensors**

Wireless legacy suit, supports 17 sensor modules at most.

**SS\_NeuronSensors**

Neuron Sensors suit with TCP/IP.

SS\_Unknown

Unknown type.

### SensorCombinationModes

Sensor combination mode.

typedef enum \_SensorCombinationModes

{

SC\_ArmOnly,

SC\_UpperBody,

SC\_FullBody,

SC\_Unknown,

}SensorCombinationModes;

**Members**

**SC\_ArmOnly**

2 nodes of left upper arm and left forearm are necessary, or, right upper arm and right forearm are necessary.

SC\_UpperBody

4 nodes of chest, hips, left upper arm and left forearm, or, chest, hips, right upper arm and right forearm are necessary.

SC\_FullBody

6 nodes of left upper leg, left leg, right upper leg, right leg, hips and chest sensor are necessary.

**SC\_Unknown**

Unknown mode.

RunningMode

Running mode of PNLib.

typedef enum \_RunningMode

{

RM\_Realtime,

RM\_RawPlaying,

RM\_Unknown,

}RunningMode;

**Members**

RM\_Realtime

Play in the window while record a motion capture.

RM\_RawPlaying

Play a raw file which was recorded a motion capture.

RM\_Unknown

Unknown mode.

SensorAcceleratorTypes

Accelerator type of sensor.

typedef enum \_SensorAcceleratorTypes

{

SA\_Range8G,

SA\_Range16G,

SA\_Range24G,

SA\_Unknown,

}SensorAcceleratorTypes;

**Members**

SA\_Range8G

Small range of accelerator type: 8G

SA\_Range16G

Medium range of accelerator type: 16G

SA\_Range24G,

Large range of accelerator type: 24G

SA\_Unknown

Unknown type

RotateOrders

Output order of rotation values around axes in BVH data.

typedef enum \_RotateOrders

{

RO\_XZY,

RO\_YXZ,

RO\_XYZ,

RO\_YZX,

RO\_ZXY,

RO\_ZYX,

RO\_Unknown,

}RotateOrders;

**Members**

RO\_XZY

Rotation value in X axis first, then Z axis, then Y axis.

RO\_YXZ

Rotation value in Y axis first, then X axis, then Z axis.

RO\_XYZ

Rotation value in X axis first, then Y axis, then Z axis.

RO\_YZX

Rotation value in Y axis first, then Z axis, then X axis.

RO\_ZXY

Rotation value in Z axis first, then X axis, then Y axis.

RO\_ZYX

Rotation value in Z axis first, then Y axis, then X axis.

RO\_Unknown

Unknown type.

OutputQuaternionTypes

Quaternion types of output stream.

typedef enum \_OutputQuaternionTypes

{

QT\_GlobalRawQuat,

QT\_GlobalBoneQuat,

QT\_LocalBoneQuat,

QT\_Unknown,

}OutputQuaternionTypes;

**Members**

QT\_GlobalRawQuat

Module quaternion in the world coordinate system.

QT\_GlobalBoneQuat

Calibrated module quaternion in the world coordinate system.

QT\_LocalBoneQuat

Quaternion relative to parent node.

QT\_Unknown,

Unknown type.

### OutputAccelerationTypes

Acceleration data types of output stream.

typedef enum \_OutputAccelerationTypes

{

AT\_ModuleRawData,

AT\_GlobalData,

AT\_Unknown,

}OutputAccelerationTypes;

Members

AT\_ModuleRawData

Module Acceleration in the world coordinate system.

AT\_GlobalData

Calibrated module acceleration in the world coordinate system.

AT\_Unknown

Unknown type.

### OutputGyroType

Gyro data types of output stream.

typedef enum \_OutputGyroType

{

GY\_ModuleRawData,

GY\_GlobalData,

GY\_Unknown,

}OutputGyroType;

**Members**

GY\_ModuleRawData

Module angular velocity in global system.

GY\_GlobalData

Calibrated module angular velocity (Bone angular velocity).

GY\_Unknown

Unknown type.

### BvhDataStreamTypes

Data types of BVH output stream.

typedef enum \_BvhDataStreamTypes

{

BO\_BinaryType,

BO\_StringType,

BO\_MatrixStringType,

BO\_Unknown,

}BvhDataStreamTypes;

**Members**

BO\_BinaryType

Binary type.

BO\_StringType

String type.

BO\_MatrixStringType

Matrix string type.

BO\_Unknown

Unknown type.

### CalculatedDataStreamTypes

Types of calculated data stream.

typedef enum \_CalculatedDataStreamTypes

{

CS\_BinaryType,

CS\_StringType,

CS\_Unknown,

}CalculatedDataStreamTypes;

**Members**

CS\_BinaryType

Binary type.

CS\_StringType

String type.

CS\_Unknown

Unknown type.

### ConstraintPoint

Bones can be edited as constraint points.

typedef enum \_ConstraintPoint

{

CP\_Hip = 0,

CP\_RightFoot = 3,

CP\_LeftFoot = 6,

CP\_RightHand = 10,

CP\_LeftHand = 14,

CP\_Unknown,

}ConstraintPoint;

Members

CP\_Hip

Hips

CP\_RightFoot

Right foot

CP\_LeftFoot

Left foot

CP\_RightHand

Right hand

CP\_LeftHand

Left hand

CP\_Unknown

Unknown type

### MagneticImmunityLevel

Magnetic immunity levels.

typedef enum \_MagneticImmunityLevel

{

MI\_Disable,

MI\_Weak,

MI\_Strong,

MI\_Unknown,

}MagneticImmunityLevel;

Members

MI\_Disable

Disable magnetic immunity function.

MI\_Weak

Low level magnetic immunity.

MI\_Strong

Strong level magnetic immunity.

MI\_Unknown

Unknown type.

### CalibrationTypes

Poses for calibration.

typedef enum \_CalibrationTypes

{

Cali\_TPose,

Cali\_APose,

Cali\_Spose,

Cali\_NPose,

Cali\_Unknown,

}CalibrationTypes;

Members

Cali\_Tpose

T pose

Cali\_Apose

A pose

Cali\_Spose

Crouching pose

Cali\_Npose

Nod pose

Cali\_Unknown

Unknown type

### PNLibVersion

Dynamic library version info.

typedef struct \_PNLibVersion

{

USHORT Major;

USHORT Minor;

USHORT Revision;

USHORT BuildNumb;

}PNLibVersion;

Members

USHORT Major

Major number

USHORT Minor

Minor number

USHORT Revision

Revision number

USHORT BuildNumb

Build number

### OutputDataVersion

BVH data stream version info.

typedef union \_OutputDataVersion

{

UINT32 \_VersionMask;

struct

{

UCHAR BuildNumb;

UCHAR Revision;

UCHAR Minor;

UCHAR Major;

};

} DATA\_VER;

Members

**\_VersionMask**

Mask of version

BuildNumb

Build number

Revision

Revision number

Minor

Subversion number

Major

Major version number

### BvhOutputBinaryHeader

Header format of BVH data in output stream.

typedef struct \_BvhOutputBinaryHeader

{

UINT16 BvhHeaderToken1;

BVH\_DATA\_VER DataVersion;

UINT32 DataCount;

BOOL WithDisp;

BOOL WithReference;

UINT32 AvatarIndex;

UCHAR AvatarName[32];

UINT32 Reserved1;

UINT32 Reserved2;

UINT16 BvhHeaderToken2;

}BvhOutputBinaryHeader;

Members

BvhHeaderToken1

Start token of package: 0xDDFF.

DataVersion

Version of community data format. e.g.: 1.0.0.2.

DataCount

Values count, 180 bytes for data without prefix nor displacements; 186 bytes for data with prefix and without displacements; 354 bytes for data without prefix and with displacements; 360 bytes for data with prefix and displacements.

WithDisp

With/without displacement.

WithReference

With/without reference bone data at first.

AvatarIndex

Avatar index.

AvatarName

Avatar name.

Reserved1

Reserved, padding bit for 64 bytes length of package.

Reserved2

Reserved, padding bit for 64 bytes length of package.

BvhHeaderToken2

End token of package: 0xEEFF.

### BvhOutputBinaryHeaderEx

Header format of compressed BVH data.

typedef struct \_BvhDataHeaderEx

{

UINT16 BvhHeaderToken1;

DATA\_VER DataVersion;

UINT32 DataCount;

BOOL WithDisp;

BOOL WithReference;

UINT32 AvatarIndex;

UCHAR AvatarName[32];

UINT32 IsCompressed;

UINT32 Reserved1;

UINT16 BvhHeaderToken2;

}BvhOutputBinaryHeaderEx;

Members

BvhHeaderToken1

Start token of package: 0xDDFF.

DataVersion

Version of community data format. e.g.: 1.0.0.2.

DataCount

Different values count with last frame if compressed.

WithDisp

With/without displacement.

WithReference

With/without reference bone data at first.

AvatarIndex

Avatar index.

AvatarName

Avatar name.

IsCompressed

Wether BVH data is compressed and compressed data check code.

Reserved1

Description of same values with last frame if compressed.

BvhHeaderToken2

End token of package: 0xEEFF.

### CalculationDataHeader

Header format of calculation data.

typedef struct \_CalculationDataHeader

{

UINT16 HeaderToken1;

DATA\_VER DataVersion;

UINT32 DataCount;

UINT32 AvatarIndex;

UCHAR AvatarName[32];

UINT32 Reserved1;

UINT32 Reserved2;

UINT32 Reserved3;

UINT32 Reserved4;

UINT16 HeaderToken2;

}CalculationDataHeader;

Members

UINT16 HeaderToken1

Start token of package: 0xDDFF.

DATA\_VER DataVersion

Version of community data format. e.g.: 1.0.0.2.

UINT32 DataCount

Values count. 17\*(16 floats) for Legacy, 59\*(16 floats) for Neuron.

UINT32 AvatarIndex

Avatar index.

UCHAR AvatarName[32]

Avatar name.

UINT32 Reserved1

Reserved, padding bit for 64 bytes length of package.

UINT32 Reserved2

Reserved, padding bit for 64 bytes length of package.

UINT32 Reserved3

Reserved, padding bit for 64 bytes length of package.

UINT32 Reserved4

Reserved, padding bit for 64 bytes length of package.

UINT16 HeaderToken2

End token of package: 0xEEFF.

### Quaternion4\_t

Quaternion

typedef struct \_Quaternion4\_t

{

float s;

float x;

float y;

float z;

}Quaternion4\_t;

Members

**s**

Rotation angle around rotation axis.

**x**

X vector to describe rotation axis.

**y**

Y vector to describe rotation axis.

**z**

Z vector to describe rotation axis.

### Vector3\_t

3D Vector with 3 float variable.

typedef struct \_Vector3\_t

{

float x;

float y;

float z;

}Vector3\_t;

Members

**x**

Coordinate in X axis.

**y**

Coordinate in Y axis.

**z**

Coordinate in Z axis.

### BoneMap

Standard bone system table.

typedef struct \_BoneMap  
{

int Index;

char Name[32];

int SensorId;

}BoneMap;

Members

Index

Bone index.

Name

Bone name.

SensorId

Sensor id bound to this bone.

### BoneDimension

Dimensions of bone, unit: meter

typedef struct \_BoneDimension

{

float Head;

float Neck;

float Body;

float ShoulderWidth;

float UpperArm;

float Forearm;

float Palm;

float HipWidth;

float UpperLeg;

float LowerLeg;

float HeelHeight;

float FootLength;

}BoneDimension;

Members

Head

Bone length of head, default: 0.18

Neck

Bone length of neck, default: 0.09

Body

Length of body, default: 0.65

ShoulderWidth

Width of shoulder, default: 0.35

UpperArm

Bone length of upper arm, default: 0.29

Forearm

Bone length of fore arm, default: 0.28

Palm

Bone length of hand, default: 0.19

HipWidth

Width of hips, default: 0.23

UpperLeg

Bone length of upper leg, default: 0.48

LowerLeg

Bone length of lower leg, default: 0.48

HeelHeight

Heel height, default: 0.05

FootLength

Foot length, default: 0.28

### ContactStatus

Status of constraint point.

typedef struct \_ContactStatus

{

ConstraintPoint Point;

PNBOOL IsEdited;

PNBOOL IsContact;

}ContactStatus;

Members

Point

Constraint point.

IsEdited

The tag whether it has been edited.

IsContact

The tag whether it is contacting.

### FrameContactData

All status of constraint points in one frame.

typedef struct \_FrameContactData

{

int FrameIndex;

ContactStatus ContactInfo[5];

}FrameContactData;

Members

FrameIndex

Index of frame data.

ContactInfo

Status of constraints.

### RawFileTime

Created time of raw file.

typedef struct \_RawFileTime

{

UCHAR Reserved;

UCHAR Second;

UCHAR Minute;

UCHAR Hour;

UCHAR Day;

UCHAR Month;

USHORT Year;

}RawFileTime;

Members

Reserved

Millisecond (0~99).

Second

The second while raw file is created.

Minute

The minute while raw file is created.

Hour

The hour while raw file is created.

Day

The day while raw file is created.

Month

The month while raw file is created.

Year

The year while raw file is created.

### RawFileAvatarInfo

One avatar information in raw file.

typedef struct \_RawFileAvatarInfo

{

char AvatarName[32];

SensorCombinationModes CombMode;

float FrontDirection[3];

BoneDimension boneDimension;

PNBOOL SensorBindingList[FULL\_BODY\_BONE\_COUNT];

}RawFileAvatarInfo;

Members

AvatarName

Avatar's name.

CombMode

Combination modes.

FrontDirection

Front direction.

boneDimension

Bone dimension.

SensorBindingList

Sensor binding list.

### RawFileInfo

Information of raw file.

typedef struct \_RawFileInfo

{

RawFileTime DateTime;

PNLibVersion LibVersion;

SensorSuitTypes SuitType;

int DataFrequency;

int TotalFrames;

int TotalTime;

int AvatarCount;

RawFileAvatarInfo\* AvatarInfoList;

}RawFileInfo;

Members

DateTime

Created time of raw file.

LibVersion

Library version of PNLib that record this file.

SuitType

Sensor suit type: Neuron or Legacy.

DataFrequency

Data acquisition frequency, Hz.

TotalFrames

Total frames in raw file.

TotalTime

Total time of playing this file, unit:second.

AvatarCount

Avatar count.

AvatarInfoList

Information list of all avatars in raw file.

### CalibrationData

Calibration data.

typedef struct \_CalibrationData

{

int AvatarIndex;

char AvatarName[64];

BoneDimension BoneDim;

Vector3\_t FaceDirection;

Vector3\_t LeftDirection;

Vector3\_t BoneDirections[FULL\_BODY\_BONE\_COUNT];

Vector3\_t BoneLeft[FULL\_BODY\_BONE\_COUNT];

Vector3\_t AccData[FULL\_BODY\_BONE\_COUNT];

}CalibrationData;

Members

AvatarIndex

Avatar index in avatar list.

AvatarName

Avatar name.

BoneDim

Bone dimension of avatar.

FaceDirection

Initial face direction at calibrated time.

LeftDirection

Initial left direction at calibrated time.

BoneDirections

Each bone direction list.

BoneLeft

Each bone left direction list.

AccData

Bone acceleration data of calibration.

### SmoothFactors

Smooth factors.

typedef struct \_SmoothFactors

{

int GlobalDisplacement;

int GlobalRotation;

int HipDisplacement;

int FeetDisplacement;

int HipRotation;

int FeetRotation;

}SmoothFactors;

Members

GlobalDisplacement

Smooth factor of displacement for all bones.

GlobalRotation

Smooth factor of rotation for all bones.

HipDisplacement

Smooth factor of displacement special for hips bone, global value would be overrided by this value.

FeetDisplacement

Smooth factor of displacement special for feet bones, global value would be overrided by this value.

HipRotation

Smooth factor of rotation special for hips bone, global value would be overrided by this value.

FeetRotation

Smooth factor of rotation special for feet bones, global value would be overrided by this value.

## Callbacks

### PNEventCalibrationProgressCallback

Calibration progress callback.

typedef void (\_\_stdcall \*PNEventCalibrationProgressCallback)  
(void\* customObject, int avatarIndex, float percent);

Members

**customObject**

User defining type.

**avatarIndex**

Avatar index.

**percent**

Percentage of avatar current calibration progress; output parameter from PNLib library.

Remarks

Fill body of this function if calibration progress is needed.

### PNEventPlayProgressCallback

Raw file playing callback.

typedef void (\_\_stdcall \*PNEventPlayProgressCallback)  
(void\* customObject, int currentFrame, int totalFrames);

Members

**customObject**

User defining type.

**currentFrame**

Current frame number of playing; output parameter from PNLib library.

**totalFrames**

Total frames of raw file; output parameter from PNLib library.

Remarks

Fill body of this function if playing status of the file is needed.

### PNEventContactNotificationCallback

Contact event callback.

typedef void (\_\_stdcall \*PNEventContactNotificationCallback)  
(void\* customObject, int boneId);

Members

**customObject**

User defining type.

**boneId**

Bone’s index; output parameter from PNLib library.

Remarks

Fill body of this function if contact notification is needed.

### PNEventRawDataParsedCallback

Parsed raw data callback.

typedef void (\_\_stdcall \*PNEventRawDataParsedCallback)  
(void\* customObject, int sensorId, Quaternion4\_t\* quat,   
Vector3\_t\* acc, Vector3\_t\* gyro);

Members

customObject

User defining type.

sensorId

Sensor’s index.

quat

Quaternion corresponding the sensor；Output parameter from PNLib library with four values s, x, y, z.

acc

Acceleration corresponding the sensor；Output parameter from PNLib library with three values x, y, z.

gyro

Gyro corresponding the sensor；Output parameter from PNLib library with three values x, y, z.

Remarks

Fill body of this function if values including quaternion, gyro and acceleration corresponding one sensor are needed.

### PNEventCalculatedStringDataCallback

Calculated frame string data callback.

typedef void (\_\_stdcall \*PNEventCalculatedStringDataCallback)  
(void\* customObject, char\* calculationData);

Members

customObject

User defining type.

calculationData

Calculated data of current frame；Output parameter from PNLib library.

Remarks

Fill body of this function if calculated data of current frame is needed. Format of calculation data refer to its definition.

### PNEventCalculatedBinaryDataCallback

Calculated frame binary data callback.

typedef void (\_\_stdcall \*PNEventCalculatedBinaryDataCallback)  
(void\* customObject, int avatarIndex,   
CalculationDataHeader\* cbp, int packLen);

Members

customObject

User defining type.

avatarIndex

Avatar index.

cbp

Calculation data header pointer.

packLen

Pack length.

Remarks

Fill body of this function if calculation data is needed as binary type of current frame. Format of calculation data refer to its definition.

### PNEventBVHStringDataBoardcastCallback

BVH string data broadcast callback.

typedef void (\_\_stdcall \*PNEventBVHStringDataBoardcastCallback)  
(void\* customObject, int avatarIndex, char\* bvhData);

Members

customObject

User defining type.

avatarIndex

Avatar index.

bvhData

Bvh data of current frame providing as string type.

Remarks

Set BvhDataStreamTypes to BO\_StringType and fill body of this function if bvh data as string type is needed; bvh data format refer to its definition.

### PNEventBVHBinaryDataBoardcastCallback

BVH binary data broadcast callback.

typedef void (\_\_stdcall \*PNEventBVHBinaryDataBoardcastCallback)  
(void\* customObject, int avatarIndex, BvhOutputBinaryPack\* bbp, int packLen);

Members

customObject

User defining type.

avatarIndex

Avatar index.

bbp

Pack pointer of BvhOutputBinary type.

packLen

Pack length.

Remarks

Set BvhDataStreamTypes to BO\_BinaryType and fill body of this function if bvh data as binary type is needed; bvh data format refer to its definition.

### PNEventBVHMatrixDataBoardcastCallback

BVH matrix string data callback.

typedef void (\_\_stdcall \*PNEventBVHMatrixDataBoardcastCallback)  
(void\* customObject, int avatarIndex, char\* matrixData);

Members

customObject

User defining type.

avatarIndex

Avatar index.

matrixData

Bvn matrix string data of current frame providing as string type.

Remarks

Set BvhDataStreamTypes to BO\_MatrixStringType and fill body of this function if bvh data as matrix string type is needed; matrix string data format refer to its definition.

### PNEventConstraintDataCallback

Constraint data callback.

typedef void (\_\_stdcall \*PNEventConstraintDataCallback)  
(void\* customObject, int avatarIndex, FrameContactData\* contactData);

Members

customObject

User defining type.

avatarIndex

Avatar index.

contactData

Pointer of FrameContactData type.

Remarks

Fill body of this function if constraint status of an avatar is needed.

### PNEventActionRecognitionDataStringStreamCallback

Action recognition string data callback.

typedef void(\_\_stdcall   
 \*PNEventActionRecognitionDataStringStreamCallback)  
(void\* customObject, int avatarIndex, char\* actionData);

Members

customObject

User defining type.

avatarIndex

Avatar index.

actionData

Pointer of action data as string type.

Remarks

Fill body of this function if action recognition data as string type of an avatar is needed.

### PNEventBodyMassVectorStringCallback

Action recognition: body mass vector.

typedef void(\_\_stdcall \*PNEventBodyMassVectorStringCallback)  
(void\* customObject, int avatarIndex, char\* data);

Members

customObject

User defining type.

avatarIndex

Avatar index.

data

Pointer of body mass data as string vector type.

Remarks

Fill body of this function if body mass data of an avatar as string vector type is needed.

### PNEventBodySwingVectorStringCallback

Action recognition: body swing vector.

typedef void(\_\_stdcall \*PNEventBodySwingVectorStringCallback)  
(void\* customObject, int avatarIndex, char\* data);

Members

customObject

User defining type.

avatarIndex

Avatar index.

data

Pointer of body swing data as string vector type.

Remarks

Fill body of this function if body swing data as string vector type of an avatar is needed.

## API reference

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Initialize and config PNLib environment \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNGetLibVersion

Get this running PNLib version number.

PNLIB\_API PNLibVersion PNGetLibVersion();

Return Value

Return a structure variable of PNLib library version. Its format is declared in “PNDataTypes.h”.

Remarks

PNLibVersion structure include four USHORT which is defined unsigned short variable type.

### PNLibInit

Initialize library.

PNLIB\_API void PNLibInit();

Remarks

Must initialize library environment by calling this function before using PNLib.

### PNGetLastErrorCode

Get error code to find more information if calling a function failed.

PNLIB\_API const PNSTATUS PNGetLastErrorCode();

Return Value

This function returns a PNLib error code. PNSTATUS is predefined as unsigned int. '0' means implementing successfully, otherwise certain error occurred when calling PNLib function.

Remarks

Can get last error code of PNLib.

### PNGetLastErrorMessage

Get last error information with windows system error code.

PNLIB\_API const char\* PNGetLastErrorMessage();

Return Value

Function returns a string message and windows system error code of PNLib last error.

Remarks

Can get last error message and windows system error code if calling a function failed.

### PNSetSensorSuitType

Set sensor suit type: Neuron or Legacy sensor type

PNLIB\_API void PNSetSensorSuitType(SensorSuitTypes modes);

Parameters

*modes*

There are two sensor suit types: Legacy and Neuron.

Remarks

Must set this property before motion capture.

### PNGetStandardBoneTable

Get standard bone system table of PNLib.

PNLIB\_API const BoneMap\* PNGetStandardBoneTable();

Return Value

Return a array point of BoneMap, the count is defined by 'FULL\_BODY\_BONE\_COUNT'

### PNSetSensorCombinationMode

Set sensor combination mode.

PNLIB\_API PNBOOL PNSetSensorCombinationMode(int avatarIndex, SensorCombinationModes mode);

Return Value

Return TRUE if set successfully, otherwise return FALSE.

Parameters

*avatarIndex*

Avatar index.

*mode*

There are three sensor combination modes: Arm only, Upper body and Full body.

Remarks

Must set this property before motion capture. PNCheckSensorBindingMode will be called in this function to check current sensor binding if it is compatible with refered mode.

### PNSetDataFolders

Set temporary folder or data export default folder.

PNLIB\_API void PNSetDataFolders(char\* appDataFolder, char\* workingFolder);

Parameters

*appDataFolder*

Calibration data will save to appData folder.

*workingFolder*

Exported data file will be saved to 'workingFolder'.English path is better. Exporting fbx file to a non-latin language path will fail.

Remarks

Calling this function will save the path in PNLib only. It will not create the folder if there isn’t.

Example

PNSetDataFolders(“appDataFolder”, “workingFolder”);

\_mkdir(“appDataFolder”);

\_mkdir(“workingFolder”);

char\* fbxname = PNExportFbxData(avatarIndex);

PNRawDataPlayStart();

Sleep(milliseconds);

PNStopExportFbxData(avatarIndex);

PNRawDataPlayStop();

### PNSetDataAcquisitionFrequency

Set data acquisition frequency. Unit: Hz

PNLIB\_API void PNSetDataAcquisitionFrequency(int freq);

Parameters

*freq*

Data acquisition frequency (unit: Hz).

Remarks

Adjust acquisition with this function befor starting to acquire data. Default value is 96Hz in PNLib.

### PNGetDataAcquisitionFrequency

Get current data acquisition frequency in PNLib.

PNLIB\_API int PNGetDataAcquisitionFrequency();

Return Value

This function returns current data acquisition frequency.

Remarks

Acquisition frequency depends on different hardware equipment.When capturing motion it will save in raw file.

### PNSetRunningMode

Set or switch run mode between real time mode and raw file playing mode.

PNLIB\_API void PNSetRunningMode(RunningMode runMode);

Parameters

*runMode*

If PNLib runs as real time capture mode, set RM\_Realtim. Or just playing raw file, set RM\_RawPlaying.

Remarks

Must set this property to rawplaying mode before opening a raw file.

### PNGetRunningMode

Get current running mode.

PNLIB\_API RunningMode PNGetRunningMode();

Return Value

This function returns current running mode.

Remarks

Can get current running mode in PNLib.

### PNLoadCalibrationData

Load calibration data.

PNLIB\_API void PNLoadCalibrationData();

Remarks

If placements of sensors do not move, this function can be called to save the calibration data of current avatar, next time the sensor configuration can be loaded directly.

### PNEnableClimbContact

Enable climb contact meanwhile disable ground contact.

PNLIB\_API void PNEnableClimbContact(int avatarIndex, PNBOOL enable);

Parameters

*avatarIndex*

Avatar index.

*enable*

Whether enabling.

Remarks

If climb is needed when capturing motion, must enable this before.

### PNResetClimbContact

Reset parameters about climb contact

PNLIB\_API void PNResetClimbContact(int avatarIndex);

Parameters

*avatarIndex*

Avatar index need reset.

Remarks

PNLib will recalculate constraint parameters when calling this function.

### PNEnableMagneticImmune

Enable magnetic immunity and set level.

If infected with surrounding ferromagnetic materials, posture accuracy of feet would be harmed. Then enable this are supposed to solve the problem.

PNLIB\_API void PNEnableMagneticImmune(int avatarIndex,  
 MagneticImmuneLevel level);

Parameters

*avatarIndex*

Avatar index of magnetic immunity need enable.

*level*

There are three magnetic immunity levels: disable, weak and strong.

Remarks

Default value is disabled. Weak level makes feet immunity and strong level makes both feet and legs.

### PNSetSpineSmoothFactors

Set smooth factors of spine.

PNLIB\_API void PNSetSpineSmoothFactors( int avatarIndex, float breastbone,   
float vertebra);

Parameters

*avatarIndex*

Avatar index.

*breastbone*

Smooth factor of Spine2 and Spine3, value range is 0~1.

*vertebra*

Smooth factor of Spine and Spine1, value range is 0~1.

### PNSetSensorAcceleratorType

Accelerator type of sensor bound to bone, modify it if using different sensor.

PNLIB\_API void PNSetSensorAcceleratorType(int boneId, SensorAcceleratorTypes type);

Parameters

*sensorId*

Sensor’s id.

*type*

Accelerator type of sensor.

Remarks

Default accelerator type of bones is SA\_Range8G. Set different range of sensors if needed.

### PNGetSensorAcceleratorType

Get accelerator type of sensor.

PNLIB\_API SensorAcceleratorTypes PNGetSensorAcceleratorType  
(int boneId);

Parameters

*sensorId*

Sensor’s id.

Return Value

This function returns current accelerator type of sensor.

Remarks

Default value is SA\_Legacy8G.

### PNReleaseScene

Release the resource of PNLib.

PNLIB\_API void PNReleaseScene();

Remarks

Calling this function to release resources when closing the PNLib.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Register and config data output callbacks \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNRegisterCalibrationProgressHandle

Register calibration progress callback.

PNLIB\_API void PNRegisterCalibrationProgressHandle(void\* customObject, PNEventCalibrationProgressCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventCalibrationProgressCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically in calibration progress.

Example

void \_stdcall \_CalibrationProgressCallback(void\* customObject, int avatarIndex,   
 double percent)

{

ControlDlg\* pControlDlg = (ControlDlg\*) customObject;

TCHAR calibrationprogress[10];

swprintf\_s(calibrationprogress, 10, \_T("%d\0"), percent);

pControlDlg->m\_static\_calibrationprogress.SetWindowText(calibrationprogress);

}

BOOL ControlDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

PNRegisterCalibrationProgressHandle(this, \_ CalibrationProgressCallback);

return TRUE;

}

### PNRegisterBvhStringDataBoardcastHandle

Register BVH string data callback handle if need deal with string type.

PNLIB\_API void PNRegisterBvhStringDataBoardcastHandle  
(void\* customObject, PNEventBVHStringDataBoardcastCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventBVHStringDataBoardcastCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically after PNLib calculate to BVH data.

Example

void \_\_stdcall \_BVHStringDataBoardcast(void\* customObject, int avatarIndex,

char\* bvhData)

{

printf("Start output no. %d avatar’s BVH data:\n%s\n", avatarIndex, bvhData);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

PNLibInit();

PNSetRunningMode(RM\_RawPlaying);

PNEnableBvhDataBoardcast(TRUE);

PNSetBvhDataBlockBoardcastType(BO\_StringType);

PNRegisterBvhStringDataBoardcastHandle(NULL, \_BVHStringDataBoardcast);

int avatarCount = PNOpenRawDataFile("RawData.raw");

PNRawDataPlayStart();

Sleep(40000);

PNRawDataPlayStop();

PNLibRelease();

return 0;

}

### PNRegisterBvhBinaryDataBoardcastHandle

Register BVH binary data callback handle if need to deal with binary type.

PNLIB\_API void PNRegisterBvhBinaryDataBoardcastHandle(void\* customObject, PNEventBVHBinaryDataBoardcastCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventBVHBinaryDataBoardcastCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically after PNLib calculate to BVH data.

Example

void \_\_stdcall \_BVHBinaryDataBoardcast(void\* customObject, int avatarIndex, struct BvhOutputBinaryHeader\* bbp, int packLen)

{

printf("Start output no. %d avatar’s BVH data:\n%s\n", avatarIndex, bvhData);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

PNLibInit();

PNSetRunningMode(RM\_RawPlaying);

PNEnableBvhDataBoardcast(TRUE);

PNSetBvhDataBlockBoardcastType(BO\_BinaryType);

PNRegisterBvhBinaryDataBoardcastHandle (NULL, \_BVHBinaryDataBoardcast);

int avatarCount = PNOpenRawDataFile("RawData.raw");

PNRawDataPlayStart();

Sleep(40000);

PNRawDataPlayStop();

PNLibRelease();

return 0;

}

### PNRegisterBvhMatrixDataBoardcastHandle

Register matrix type of BVH data callback handle if need get matrix string data.

PNLIB\_API void PNRegisterBvhMatrixDataBoardcastHandle(void\* customObject, PNEventBVHMatrixDataBoardcastCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventBVHMatrixDataBoardcastCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically after PNLib calculate to BVH data.

Example

void \_\_stdcall \_BVHMatrixDataBoardcast(void\* customObject, int avatarIndex,

char\* matrixData)

{

printf("Start output no. %d avatar’s Matrix data:\n%s\n", avatarIndex, matrixData);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

PNLibInit();

PNSetRunningMode(RM\_RawPlaying);

PNEnableBvhDataBoardcast(TRUE);

PNSetBvhDataBlockBoardcastType(BO\_MatrixStringType);

PNRegisterBvhMatrixDataBoardcastHandle(NULL, \_BVHMatrixDataBoardcast);

int avatarCount = PNOpenRawDataFile("RawData.raw");

PNRawDataPlayStart();

Sleep(40000);

PNRawDataPlayStop();

PNLibRelease();

return 0;

}

### PNSetBvhDataBlockBoardcastType

Change output type of BVH data. Must register the relevant callback handle before change to a certain one.

PNLIB\_API void PNSetBvhDataBlockBoardcastType(BvhDataStreamTypes type);

Parameters

*type*

Data types of BVH output stream: Binary, String and MatrixString.

Remarks

Must specify the corresponding type if registering one type of bvh broadcast callback function.

Example

PNEnableBvhDataBoardcast(TRUE);

PNSetBvhDataBlockBoardcastType(BO\_MatrixStringType);

PNRegisterBvhMatrixDataBoardcastHandle(NULL, \_BVHMatrixDataBoardcast);

### PNEnableBvhDataBoardcast

Disable BVH callback if it is not needed as it occupies lots of CPU.

PNLIB\_API void PNEnableBvhDataBoardcast(PNBOOL isEnable);

Parameters

*isEnable*

Whether enable the bvh broadcast.

Remarks

Must enable it if need bvh broadcast callback function.

Example

PNEnableBvhDataBoardcast(TRUE);

PNSetBvhDataBlockBoardcastType(BO\_MatrixStringType);

PNRegisterBvhMatrixDataBoardcastHandle(NULL, \_BVHMatrixDataBoardcast);

### PNSetCalculatedDataBlockBoardcastType

Change output type of Calculated data.

PNLIB\_API void PNSetCalculatedDataBlockBoardcastType  
(CalculatedDataStreamTypes type);

Parameters

*type*

Data types of Calculated output stream: Binary and String.

Remarks

Must specify the corresponding type if registering one type of calculated broadcast callback function.

### PNRegisterCalculatedStringDataBoardcastHandle

Register calculation data callback handle.

PNLIB\_API void PNRegisterCalculatedStringDataBoardcastHandle  
(void\* customObject, PNEventCalculatedStringDataCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventCalculatedStringDataCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically after PNLib calculate to calculated data.

### PNRegisterCalculatedBinaryDataBoardcastHandle

Register calculation data callback handle.

PNLIB\_API void PNRegisterCalculatedBinaryDataBoardcastHandle  
(void\* customObject, PNEventCalculatedBinaryDataCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventCalculatedBinaryDataCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically after PNLib calculate to calculated data.

### PNSetCalculatedQuaternionDataType

Change quaternion type in calculation frame data.

PNLIB\_API void PNSetCalculatedQuaternionDataType  
(OutputQuaternionTypes type);

Parameters

*type*

Quaternion types of output stream: Global Raw Quaternion, Global Bone Quaternion and Local Bone Quaternion.

### PNSetCalculatedAccelerationDataType

Change Acceleration type in calculation frame data.

PNLIB\_API void PNSetCalculatedAccelerationDataType  
(OutputAccelerationTypes type);

Parameters

*type*

Acceleration data types of output stream: Module Raw Data and Global Data.

### PNSetCalculatedGyroDataType

Change Gyro type in calculation frame data.

PNLIB\_API void PNSetCalculatedGyroDataType(OutputGyroType type);

Parameters

*type*

Gyro data types of output stream: Module Raw Data and Global Data.

### PNEnableCalculationDataBoardcast

Enable/disable calculation data callback.

PNLIB\_API void PNEnableCalculationDataBoardcast(PNBOOL enable);

Parameters

*enablbe*

Whether enable the calculation data broadcast.

Remarks

Must enable it if calculation data broadcast callback function is needed.

### PNRegisterRawDataPlayingProgressHandle

Register raw data playing progress callback.

PNLIB\_API void PNRegisterRawDataPlayingProgressHandle  
(void\* customObject, PNEventPlayProgressCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventPlayProgressCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically in raw data playing progress.

Example

void \_stdcall \_PlayProgressCallback(void\* customObject, int currentFrame, int totalFrames)

{

ControlDlg\* pControlDlg = (ControlDlg\*) customObject;

If(currentFrame == totalFrames)

pControlDlg->m\_static\_finish.SetWindowText(“Endplay”);

}

BOOL ControlDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

PNRegisterRawDataPlayingProgressHandle(this, \_PlayProgressCallback);

return TRUE;

}

### PNRegisterPlayingRawDataParsedHandle

Register raw data parsed data callback.

PNLIB\_API void PNRegisterPlayingRawDataParsedHandle(void\* customObject, PNEventRawDataParsedCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventRawDataParsedCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically in raw data parsed progress.

### PNRegisterContactNotification

Register contact event callback.

PNLIB\_API void PNRegisterContactNotification(void\* customObject, PNEventContactNotificationCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventContactNotificationCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically in contact event. Before enabling this function, enable the contact event notification first.

### PNRegisterContactEditCallback

Register contact edit callback.

PNLIB\_API void PNRegisterContactEditCallback(void\* customObject, PNEventConstraintDataCallback editHandler);

Parameters

*customObject*

User can defined any type pointer needed.

*editHandler*

A function pointer of PNEventConstraintDataCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically in contact edit.

### PNEnableContactNotification

Enable/disable contact event notification.

PNLIB\_API void PNEnableContactNotification(PNBOOL enable);

Parameters

*enable*

Whether enable the contact event notification.

Remarks

Must enable it if contact event notification callback function is needed.

### PNRegisterActionRecognizationStringDataBoardcastHandle

Register action recognition event callback.

PNLIB\_API void PNRegisterActionRecognizationStringDataBoardcastHandle  
(void\* customObject,   
PNEventActionRecognizationDataStringStreamCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventActionRecognizationDataStringStreamCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically in action recognization event.

### PNRegisterBodyMassVectorStringCallback

Register body mass vector of action recognition callback.

PNLIB\_API void PNRegisterBodyMassVectorStringCallback  
(void\* customObject, PNEventBodyMassVectorStringCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventBodyMassVectorStringCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically in action recognization event if body mass vector is needed.

### PNRegisterBodySwingVectorStringCallback

Register body swing vector of action recognition callback.

PNLIB\_API void PNRegisterBodySwingVectorStringCallback   
(void\* customObject, PNEventBodySwingVectorStringCallback handle);

Parameters

*customObject*

User can defined any type pointer needed.

*handle*

A function pointer of PNEventBodySwingVectorStringCallback type.

Remarks

Register this function when program initializes. The callback function defined with the prescribed format will be called automatically in action recognization event if the body swing vector is needed.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Manage avatars \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNCreateAvatar

Create an avatar in scene.

PNLIB\_API int PNCreateAvatar();

Return Value

Return current avatar index.

Remarks

Cannot see this avatar in screen if it has no motion data.

### PNRemoveAvatar

Delete an avatar from scene.

PNLIB\_API int PNRemoveAvatar(int avatarIndex);

Parameters

*avatarIndex*

Avatar index need delete.

Return Value

Return remainder avatar count.

Remarks

It will not delete the avatar from data file unless saving file.

### PNGetAvatarCount

Get total avatars count in current scene.

PNLIB\_API int PNGetAvatarCount();

Return Value

Return current avatar count in PNLib.

Remarks

Include avatar that created.

### PNSetAvatarName

Set a name for avatar.

PNLIB\_API void PNSetAvatarName(int avatarIndex, char\* name);

Parameters

*avatarIndex*

Avatar index.

*name*

Avatar name corresponding to the index.

### PNGetAvatarName

Get avatar's name.

PNLIB\_API void PNGetAvatarName(int avatarIndex, char\* name,   
int buffLen);

Parameters

*avatarIndex*

Avatar index.

*name*

Head pointer of buffer saving the name.

*buffLen*

Buffer capability.

Remarks

Name is an output parameter and length of which must be equal to or smaller than the actual name buffer, otherwise an incomplete name will be got.

### PNSetBoneDimensions

Set avatar's bone dimensions.

PNLIB\_API void PNSetBoneDimensions(int avatarIndex,   
BoneDimension\* dimensions);

Parameters

*avatarIndex*

Avatar index.

*dimensions*

Pointer of BoneDimension type.

Remarks

Define a BoneDimension structure and input its pointer to PNLib.

### PNGetBoneDimensions

Get avatar's bone dimensions.

PNLIB\_API void PNGetBoneDimensions(int avatarIndex,   
BoneDimension\* dimsBuffer);

Parameters

*avatarIndex*

Avatar index.

*dimsBuff*

Pointer of BoneDimension type used to save bone dimensions from PNLib.

Remarks

A default bone dimension will be got if creating an avatar and not set its bone dimensions.

### PNGetBoneLength

Get avatar's bone length.

PNLIB\_API float PNGetBoneLength(int avatarIndex, int boneIndex);

Parameters

*avatarIndex*

Avatar index.

*boneIndex*

Bone index in Standard Bone Table.

Return Value

Return the length of the bone corresponding to the bone index.

Remarks

59 bones’ length can be got in Neuron with fingers. 21 bones’ length in Legacy without fingers. Call this function to get their lengths.

Fingers will be scaled by palm length. Others are calculated by the BoneDimension structure as below:

|  |  |  |
| --- | --- | --- |
| **Bone Index** | **Bone Name** | **Calculation** |
| 0 | Hips | Body \* 1/5 |
| 1 | RightUpLeg | UpperLeg |
| 2 | RightLeg | LowerLeg |
| 3 | RightFoot | FootLength |
| 4 | LeftUpLeg | UpperLeg |
| 5 | LeftLeg | LowerLeg |
| 6 | LeftFoot | FootLength |
| 7 | RightShoulder | ShoulderWidth \* 1/2 |
| 8 | RightArm | UpperArm |
| 9 | RightForeArm | Forearm |
| 10 | RightHand | Palm |
| 11 | LeftShoulder | ShoulderWidth \* 1/2 |
| 12 | LeftArm | UpperArm |
| 13 | LeftForeArm | Forearm |
| 14 | LeftHand | Palm |
| 15 | Head | Head |
| 16 | Neck | Neck |
| 17 | Spine3 | Body \* 1/5 |
| 18 | Spine2 | Body \* 1/5 |
| 19 | Spine1 | Body \* 1/5 |
| 20 | Spine | Body \* 1/5 |

### PNBindSensor

Binding a sensor to bone. Return FALSE if failed.

PNLIB\_API PNBOOL PNBindSensor(int avatarIndex, int boneIndex,   
int sensorId);

Return Value

Return TRUE if binding succeeds, otherwise return FALSE.

Parameters

*avatarIndex*

Avatar index.

*boneIndex*

Bone index.

*sensorId*

Sensor id.

Remarks

Ensure right avatarIndex, boneIndex and sensorId or the data accuracy might be harmed.

### PNRemoveSensor

Remove sensor from referred bone. Return FALSE if failed.

PNLIB\_API PNBOOL PNRemoveSensor(int avatarIndex, int boneIndex);

Return Value

Return TRUE if removing succeeds, otherwise return FALSE.

Parameters

*avatarIndex*

Avatar index, start from 0.

*boneIndex*

Bone index, start from 0.

Remarks

The sensorId of this boneIndex will be 0 after calling this function.

### PNIsBindingSensor

Check the referred bone if is bound sensor.

PNLIB\_API PNBOOL PNIsBindingSensor(int avatarIndex, int boneIndex);

Return Value

Return TRUE if binding currently, otherwise return FALSE.

Parameters

*avatarIndex*

Avatar index, start from 0.

*boneIndex*

Bone index, start from 0.

### PNCheckSensorBindingMode

Check the current sensor binding if is suitable with the referred sensor combination mode.

PNLIB\_API PNBOOL PNCheckSensorBindingMode(int avatarIndex, SensorCombinationModes mode);

Return Value

Return TRUE if suitable, otherwise return FALSE.

Parameters

*avatarIndex*

Avatar index, start from 0.

*mode*

Sensor combination mode.

Remarks

For SC\_ArmOnly mode, 2 nodes of left upper arm and left forearm are necessary, or, right upper arm and right forearm are necessary.

For SC\_UpperBody mode, 4 nodes of chest, hips, left upper arm and left forearm are necessary, or, chest, hips, right upper arm and right forearm are necessary.

For SC\_FullBody mode, 6 nodes of left upper leg, left leg, right upper leg, right leg, hips and chest sensor are necessary.

### PNResetBoneMapping

Reset sensors binding to default bone map. Return FALSE if failed.

PNLIB\_API PNBOOL PNResetBoneMapping(int avatarIndex);

Return Value

Return TRUE if mapping succeeds, otherwise return FALSE.

Parameters

*avatarIndex*

Avatar index.

Remarks

All sensorId of this avatar’s bone will be 0 after calling this function.

### PNGetBoneName

Get bone name by avatar index and bone index.

PNLIB\_API char\* PNGetBoneName(int avatarIndex, int boneIndex);

Return Value

Get bone name corresponding with a bone of avatar.

Parameters

*avatarIndex*

Avatar index.

*boneIndex*

Bone index.

Remarks

All bone name refer to Standard Bone Table.

### PNGetBoneNameBySensorId

Get bone name by avatar index and sensor id.

PNLIB\_API char\* PNGetBoneNameBySensorId(int avatarIndex, int sensorId);

Return Value

Get bone name corresponding with a bound sensor id of avatar.

Parameters

*avatarIndex*

Avatar index.

*sensorId*

Sensor id starting from 1.

Example

PNBOOL bindresult = PNBindSensor(avatarIndex, boneindex, sensorid);

Char\* actualname = PNGetBoneNameBySensorId(avatarIndex, sensorid);

### PNGetSensorId

Get sensor id by avatar index and bone index.

PNLIB\_API int PNGetSensorId(int avatarIndex, int boneIndex);

Return Value

Get bound sensor id corresponding with bone index of avatar.

Parameters

*avatarIndex*

Avatar index.

*boneIndex*

Bone index starting from 0.

Remarks

Get sensor ID based on bone index, starting from 1. Return 0 means no sensor bound on the referred bone, -1 means error occurred.

### PNGetBoneIndexBySensorId

Get bone index by avatar index and sensor id.

PNLIB\_API int PNGetBoneIndexBySensorId(int avatarIndex, int sensorId);

Return Value

Bone index.

Parameters

*avatarIndex*

Avatar index.

*sensorId*

Sensor id.

Remarks

Return -1 if this sensor was not bound to a bone.

### PNGetHipWidth

Get hips width of avatar.

PNLIB\_API float PNGetHipWidth(int avatarIndex);

Return Value

Return a float type variable indicating hips width.

Parameters

*avatarIndex*

Avatar index.

Remarks

Return 0 if invalid avatar index. The default value is 0.23f. Set it with PNSetBoneDimensions.

### PNGetHipHeight

Get hips height of avatar, the height is the distance from hips to ground.

PNLIB\_API float PNGetHipHeight(int avatarIndex);

Return Value

Return a float type variable indicating the length of hips from ground.

Parameters

*avatarIndex*

Avatar index.

Remarks

Return 0 if invalid avatar index. The default value is 0.96f which is the length of UpperLeg plus LowerLeg. Set it with PNSetBoneDimensions.

### PNGetShoulderWidth

Get shoulder width of avatar.

PNLIB\_API float PNGetShoulderWidth(int avatarIndex);

Return Value

Return a float type variable indicating the shoulder width.

Parameters

*avatarIndex*

Avatar index.

Remarks

Return 0 if invalid avatar index. The default value is 0.35f. Set it with PNSetBoneDimensions.

### PNGetHeelHeight

Get heel height of avatar.

PNLIB\_API float PNGetHeelHeight(int avatarIndex);

Return Value

Return a float type variable indicating the heel height.

Parameters

*avatarIndex*

Avatar index.

Remarks

Return 0 if invalid avatar index. The default value is 0.05f. Set it with PNSetBoneDimensions.

### PNGetInitiationDirection

Get direction of avatar at time of calibration.

PNLIB\_API void PNGetInitiationDirection(int avatarIndex,   
Vector3\_t\* zd);

Parameters

*avatarIndex*

Avatar index.

*zd*

Vector3 type pointer of direction.

### PNGetInitiationLeftDirection

Get the left direction of avatar at time of calibration.

PNLIB\_API void PNGetInitiationLeftDirection(int avatarIndex,   
Vector3\_t\* xd);

Parameters

*avatarIndex*

Avatar index.

*xd*

Vector3 type pointer of left direction.

### PNCanCalibratePose

Check whether a pose of calibration is necessary based on current sensor bound.

PNLIB\_API PNBOOL PNCanCalibratePose(int avatarIndex,   
CalibrationTypes type);

Return Value

Return TRUE if this calibration is necessary, otherwise return FALSE.

Parameters

*avatarIndex*

Avatar index.

*type*

Calibration Type.

Remarks

If left upper arm and left forearm or right upper arm and right fore arm are bound, A pose and T pose are necessary.

If left upper leg, left leg, right upper leg, right leg, hips and spine sensor are bound, S pose is necessary.

### PNGetCalibrationData

Get calibration data.

PNLIB\_API void PNGetCalibrationData(int avatarIndex,   
struct CalibrationData\* data);

Parameters

*avatarIndex*

Avatar index.

*data*

Pointer of CalibrationData struct data.

### PNSetCalibrationData

Set calibration data.

PNLIB\_API void PNSetCalibrationData(int avatarIndex,   
struct CalibrationData\* data);

Parameters

*avatarIndex*

Avatar index.

*data*

Pointer of CalibrationData struct data.

### PNClearIntegralState

Clear integral state.

PNLIB\_API void PNClearIntegralState(int avatarIndex);

Parameters

*avatarIndex*

Avatar index.

Remarks

An error code and message will be got if invalid avatar index is sent in.

### PNSetDataOutputFrequencyRatio

Set data output frequency ratio.

PNLIB\_API void PNSetDataOutputFrequencyRatio(int ratio);

Parameters

*ratio*

Ratio of data output frequency.

Remarks

Zero or negative values will be set failure and send error code. The actual output frequency is the set frequency with PNSetDataAcquisitionFrequency dividing by ratio, such as:

|  |  |  |  |
| --- | --- | --- | --- |
| frequency | ratio | formula | result |
| 30 | 1 | 30/1 | 30 |
| 48 | 2 | 48/2 | 24 |
| 60 | 4 | 60/4 | 15 |
| 96 | 8 | 96/8 | 12 |

### PNGetDataOutputFrequencyRatio

Get data output frequency ratio.

PNLIB\_API int PNGetDataOutputFrequencyRatio();

Return Value

Return ratio of data output frequency.

Remarks

Default value is 1 after PNLib initialize.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Calculation parameters settings \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNGetKalmanParams

Get kalman parameters.

PNLIB\_API struct KalmanParams\* PNGetKalmanParams(int avatarIndex);

Return Value

Pointer of KalmanParams struct.

Parameters

avatarIndex

Avatar index.

### PNSetKalmanParams

Set kalman parameters.

PNLIB\_API void PNSetKalmanParams(int avatarIndex,   
struct KalmanParams params);

Parameters

*avatarIndex*

Avatar index.

*params*

KalmanParams struct.

### PNResetKalmanParams

Reset kalman parameters.

PNLIB\_API void PNResetKalmanParams(int avatarIndex);

Parameters

*avatarIndex*

Avatar index.

### PNSetPDamping

Set damping coefficient of preventing P diffuse.

PNLIB\_API void PNSetPDamping(int avatarIndex, float pdamping);

Parameters

*avatarIndex*

Avatar index.

*pdamping*

Float type value.

### PNGetPDamping

Get damping coefficient of preventing P diffuse.

PNLIB\_API float PNGetPDamping(int avatarIndex);

Return Value

Return current damping coefficient.

Parameters

*avatarIndex*

Avatar index.

### PNSetJointStiffness

Set joint stiffness.

PNLIB\_API void PNSetJointStiffness(int avatarIndex, float percent);

Parameters

*avatarIndex*

Avatar index.

*percent*

Percent of joint stiffness.

### PNSetStepStiffness

Set step stiffness.

PNLIB\_API void PNSetStepStiffness(int avatarIndex, float percent);

Parameters

*avatarIndex*

Avatar index.

*percent*

Percent of step stiffness.

### PNSetStepConstraint

Set step constraint.

PNLIB\_API void PNSetStepConstraint(int avatarIndex, float percent);

Parameters

*avatarIndex*

Avatar index.

*percent*

Percent of step constraint.

### PNGetStiffnessPercent

Get stiffness percent.

PNLIB\_API void PNGetStiffnessPercent(int avatarIndex,   
float\* jointStiffnessPercent,   
float\* stepStiffnessPercent,   
float\* stepConstraintPercent);

Parameters

*avatarIndex*

Avatar index.

*jointStiffnessPercent*

Float type pointer to save joint stiffness percent.

*stepStiffnessPercent*

Float type pointer to save step stiffness percent.

*stepConstraintPercent*

Float type pointer to save step constraint percent.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Smooth settings \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNEnableSmoothFilter

Enable smooth filter.

PNLIB\_API void PNEnableSmoothFilter(int avatarIndex, PNBOOL enable);

Parameters

*avatarIndex*

Avatar index.

*enable*

BOOL type value. Set TRUE if enable smooth filter, otherwise set FALSE. It is disabled by default.

### PNIsSmoothFilterEnabled

Check whether smooth filter is enabled.

PNLIB\_API PNBOOL PNIsSmoothFilterEnabled(int avatarIndex);

Return Value

Return TRUE if smooth filter is enabled, otherwise return FALSE.

Parameters

*avatarIndex*

Avatar index.

### PNSetSmoothFactor

Set smooth factor.

PNLIB\_API void PNSetSmoothFactor(int avatarIndex,   
SmoothFactors smoothFactors);

Parameters

*avatarIndex*

Avatar index.

*smoothFactors*

SmoothFactors struct.

Remarks

Need set smooth factor if smooth filter is enabled.

### PNGetSmoothFactor

Get current smooth factor in PNLib.

PNLIB\_API void PNGetSmoothFactor(int avatarIndex,   
SmoothFactors\* smoothFactors);

Parameters

*avatarIndex*

Avatar index.

*smoothFactors*

Pointer of SmoothFactors struct.

### PNEnableFootLock

Enable/disable foot lock while certain foot contacts ground.

PNLIB\_API void PNEnableFootLock(PNBOOL enable);

Parameters

*enable*

BOOL type value. Set TRUE if enable foot lock, otherwise set FALSE. It is enabled by default.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Manage drawing scene \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNCreateBvhPlayer

Create bvh player.

PNLIB\_API void\* PNCreateBvhPlayer(BVHWindowContainerRef hWndParent);

Return Value

Window handle of bvhplayer. Null if creation fails.

Parameters

*hWndParent*

HWND of BVH Window to show in.

Remarks

Create a bvh player in memory with this function. Call PNBvhPlayerResizeToParent if need to show it in window.

### PNSetBvhPlayerCameras

Split and duplicate drawing scene.

PNLIB\_API void PNSetBvhPlayerCameras(int cameras);

Parameters

*cameras*

Number of camera, most is 4.

### PNEnableBvhPlayerCameraBind

Bind/unbind camera to an avatar.

PNLIB\_API void PNEnableBvhPlayerCameraBind(int cameraIndex,   
int avatarIndex,   
PNBOOL enableBind);

Parameters

*cameraIndex*

Camera index, start from 0.

*avatarIndex*

Avatar index.

*enableBind*

Set TRUE to bind or set FALSE to unbind.

### PNBvhPlayerResizeToParent

Resize drawing window to fill the container.

PNLIB\_API void PNBvhPlayerResizeToParent();

Remarks

To show bvh player in window, must call PNCreateBvhPlayer firstly.

### PNCloseBvhPlayer

Release drawing scene.

PNLIB\_API void PNCloseBvhPlayer();

Remarks

Call this function to close bvh player.

### PNEnableMassShowing

Enable/disable mass showing for some avatar.

PNLIB\_API void PNEnableMassShowing(int avatarIndex, PNBOOL enable);

Parameters

*avatarIndex*

Avatar index.

*enable*

Set TRUE if showing mass otherwise set FALSE.

### PNEnableRendering

Enable/disable rendering.

PNLIB\_API void PNEnableRendering(PNBOOL enable);

Parameters

*enable*

Set TRUE if rendering otherwise set FALSE.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Realtime data interface \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNPushData

Push raw data into PNLib.

PNLIB\_API void PNPushData(unsigned char\* data);

Parameters

*data*

Pointer of one data package.

Remarks

Data entrance, if avatar number is unknown, push data that received from serial port or network to PNLib in a single package method.

### PNPushDataForAvatar

Push avatar's raw data into PNLib.

PNLIB\_API void PNPushDataForAvatar(int avatarIndex,   
unsigned char\* data);

Parameters

*avatarIndex*

Avatar index.

*data*

Pointer of single data package for corresponding avatar.

Remarks

If corresponding avatar index is known, push each data to PNLib with this function.

### PNEnableLostDataFitting

Enable/disable fitting of lost data.

PNLIB\_API void PNEnableLostDataFitting(PNBOOL enable);

Parameters

*enable*

Set TRUE to enable lost data fitting otherwise set FALSE.

### PNClearCalibrationBufferedData

Clear the buffered data used to calibration.

PNLIB\_API void PNResetCalibrationSteps(int avatarIndex);

Remarks

Prior to a new calibration, it is required to clear the previous buffered data by this function.

### PNCalibrateAvatar

Start a calibrating action of an avatar.

PNLIB\_API void PNCalibrateAvatar(int avatarIndex, CalibrationTypes type);

Parameters

*avatarIndex*

Avatar index.

*type*

Calibration type.

### PNCalibrateAllAvatars

Start a calibrating action for all avatar.

PNLIB\_API void PNCalibrateAllAvatars(CalibrationTypes type);

Parameters

*type*

Calibration type.

### PNGetSensorReceivingStatus

Get receiving percentage of sensor data.

PNLIB\_API int PNGetSensorReceivingStatus(int avatarIndex,   
float bufferForPercentData [FULL\_BODY\_BONE\_COUNT]);

Return Value

Return current sensor number.

Parameters

*avatarIndex*

Avatar index.

*bufferForPecentData*

Float type array buffer for percent data.

Remarks

FULL\_BODY\_BONE\_COUNT is 167 defined in PNDataTypes.h.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Raw file interface \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNGetRawFileInfo

Get the referred raw file information. Such as record time, sensor type, freq, avatar count, etc.

PNLIB\_API RawFileInfo\* PNGetRawFileInfo(char\* filename);

Return Value

Return a pointer of RawFileInfo type.

Parameters

*Filename*

File path.

### PNOpenRawDataFile

Open raw file to replay the captured motion data, return the avatar number in this motion data file.

PNLIB\_API int PNOpenRawDataFile(char\* filename);

Return Value

Return avatar number of current file if open successfully, and return 0 if failed.

Parameters

*filename*

File path.

Example

PNLibInit();

PNSetRunningMode(RM\_RawPlaying);

int avatarCount = PNOpenRawDataFile("RawDataFile.raw");

### PNRawDataPlayGetTotalFrames

Get total frames in opened raw file.

PNLIB\_API unsigned long PNRawDataPlayGetTotalFrames();

Return Value

Return total frames of current file.

### PNGetSensorSuitType

Get sensor suit type in opened raw file.

PNLIB\_API SensorSuitTypes PNGetSensorSuitType();

Return Value

Return current sensor suit type of file.

### PNGetSensorCombinationMode

Get node combination mode.

PNLIB\_API SensorCombinationModes PNGetSensorCombinationMode   
(int avatarIndex);

Return Value

Return sensor combination mode of corresponding avatar.

Parameters

*avatarIndex*

Avatar index.

Remarks

Default value is SC\_Unknown.

### PNRawDataPlaySetPlayingPosition

Set playing position.

PNLIB\_API BOOL PNRawDataPlaySetPlayingPosition (int pos);

Return Value

Return TRUE if set successfully, otherwise return FALSE.

Parameters

*pos*

Frame index to play. Start from 0. Biggest number is totalFrames subtracting 1.

### PNRawDataPlayGetCurrentPlayingPosition

Get current playing position.

PNLIB\_API unsigned long PNRawDataPlayGetPlayingPosition();

Return Value

Return current playing frame index which start from 0.

### PNRawDataPlaySetSpeed

Set playing speed ratio.

PNLIB\_API void PNRawDataPlaySetSpeed(float ratio);

Parameters

*ratio*

Current playing speed ratio. The more the faster.

### PNRawDataPlayStart

Start playing data.

PNLIB\_API void PNRawDataPlayStart();

### PNRawDataPlayPause

Raw data playing pause.

PNLIB\_API void PNRawDataPlayPause();

### PNRawDataPlayStop

Raw data play stop.

PNLIB\_API void PNRawDataPlayStop();

### PNRawDataPlayEnableReversePlaying

Enable reverse playing.

PNLIB\_API void PNRawDataPlayEnableReversePlaying(BOOL enable);

Parameters

*enable*

A BOOL type variable. Set TRUE to enable it, otherwise set FALSE.

### PNRawDataPlaySetToPrev

Set to previous frame.

PNLIB\_API unsigned long PNRawDataPlaySetToPrev();

Return Value

Return previous frame index.

### PNRawDataPlaySetToNext

Set to next frame.

PNLIB\_API unsigned long PNRawDataPlaySetToNext();

Return Value

Return next frame index.

### PNCloseRawDataFile

Close raw data file and lean space of raw file.

PNLIB\_API void PNCloseRawDataFile();

Remarks

Close current file before open another.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Constraint editing \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNEditContact

Edit constraint contact.

PNLIB\_API PNSTATUS PNEditContact(int avatarIndex, int frameIndex,   
ConstraintPoint point, PNBOOL isContact);

Return Value

Return error code if failed. Get last error message with PNGetLastErrorMessage.

Parameters

*avatarIndex*

Avatar index.

*frameIndex*

Frame index.

*point*

Enumeration variable of ConstraintPoint type.

*isContact*

BOOL type variable means whether contacting.

### PNGetContactStatus

Get constraint contact status.

PNLIB\_API PNSTATUS PNGetContactStatus (int avatarIndex, int frameIndex,   
ConstraintPoint point, BOOL\* isContact);

Return Value

Return error code. Return 0 if success, otherwise return corresponding error code.

Parameters

*avatarIndex*

Avatar index.

*frameIndex*

Frame index.

*point*

Enumeration variable of ConstraintPoint type.

*isContact*

Pointer of BOOL type send constraint status corresponding constraint point for avatar.

### PNBatchEditContact

Edit a batch of constraint contact.

PNLIB\_API PNSTATUS PNBatchEditContact(int avatarIndex,   
int startFrameIndex, int endFrameIndex, ConstraintPoint point, BOOL isContact);

Return Value

Return error code. Return 0 if success, otherwise return corresponding error code.

Parameters

*avatarIndex*

Avatar index.

*startFrameIndex*

Start frame index editing.

*endFrameIndex*

End frame index of editing frame.

*point*

Enumeration variable of ConstraintPoint type.

*isContact*

BOOL type variable means contacting status.

### PNBatchResetContactEditStatus

PNLIB\_API PNSTATUS PNBatchResetContactEditStatus (int avatarIndex,   
int startFrameIndex, int endFrameIndex, ConstraintPoint point);

Return Value

Return error code. Return 0 if success, otherwise return corresponding error code.

Parameters

*avatarIndex*

Avatar index.

*startFrameIndex*

Start frame index editing.

*endFrameIndex*

End frame index of editing frame.

*point*

Enumeration variable of ConstraintPoint type.

### PNResetContactEditStatus

Cancel contact edit of some contact point of some frame.

PNLIB\_API PNSTATUS PNResetContactEditStatus(int avatarIndex,   
int frameIndex, ConstraintPoint point);

Return Value

Return error code. Return 0 if success, otherwise return corresponding error code.

Parameters

*avatarIndex*

Avatar index.

*frameIndex*

Frame index.

*point*

Enumeration variable of ConstraintPoint type.

### PNFeetConstraintOptimization

Optimization feet constraint with the specified level.

PNLIB\_API void PNFeetConstraintOptimization(int avatarIndex,   
unsigned int level);

Parameters

*avatarIndex*

Avatar index.

*level*

Unsigned int type. 0 means reset to no optimized status. 10 is the max level can be used.

Remarks

Reset level will reset all the constraint status include result edited.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Data export operations \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNExportRawData

Start to export raw data to file.

PNLIB\_API char\* PNExportRawData();

Return Value

Return a pointer of char type pointing file name saved.

Remarks

Export a separate raw file for each avatar by default.

### PNStopExportRawData

Stop to export raw data to file.

PNLIB\_API void PNStopExportRawData();

### PNExportRawDataTxt

Start to export raw data about one sensor of one avatar.

PNLIB\_API char\* PNExportRawDataTxt(int avatarIndex, int sensorId);

Return Value

Return a pointer of char type pointing file name saved.

Remarks

Export a raw file about one sensor of one avatar.

### PNStopExportRawDataTxt

Stop to export raw data about one sensor of one avatar.

PNLIB\_API void PNStopExportRawDataTxt(int avatarIndex);

### PNExportCalculationData

Start to export calculation data to file.

PNLIB\_API char\* PNExportCalculationData(int avatarIndex);

Return Value

Return a pointer of char type pointing file name saved.

Parameters

*avatarIndex*

Avatar index.

### PNStopExportCalculationData

Stop to export calculation data to file.

PNLIB\_API void PNStopExportCalculationData(int avatarIndex);

Parameters

*avatarIndex*

Avatar index.

### PNExportBvhData

Start to export BVH data.

PNLIB\_API char\* PNExportBvhData(int avatarIndex);

Return Value

Return a pointer of char type pointing file name saved.

Parameters

*avatarIndex*

Avatar index.

### PNStopExportBvhData

Stop to export BVH data.

PNLIB\_API void PNStopExportBvhData(int avatarIndex);

Return Value

Return a pointer of char type pointing file name saved.

Parameters

*avatarIndex*

Avatar index.

### PNExportFbxData

Start to export fbx data.

PNLIB\_API char\* PNExportFbxData(int avatarIndex);

Return Value

Return a pointer of char type pointing file name saved.

Parameters

*avatarIndex*

Avatar index.

Remarks

Exporting to a Chinese or special symbol path will be failed.

### PNStopExportFbxData

Stop to export fbx data.

PNLIB\_API void PNStopExportFbxData(int avatarIndex);

Parameters

*avatarIndex*

Avatar index.

Remarks

Exporting to a Chinese or special symbol path will be failed.

### PNSetBvhDataFormat

Set bvh data format.

PNLIB\_API void PNSetBvhDataFormat(BOOL isWithDisp,   
enum RotateOrders order);

Parameters

*isWithDisp*

BOOL type variable means whether is with displacement. Set TRUE if is, otherwise set FALSE.

*order*

Enum variable of RotateOrders type.

### PNSetBvhDataWithReference

With/without reference before export bvh data.

PNLIB\_API void PNSetBvhDataWithReference(BOOL withReference);

Parameters

*withReference*

With prefixion or not. Set TRUE if is, otherwise set FALSE.

### PNBvhBinaryDataOutputIsCompression

BVH data output compression flag.

PNLIB\_API void PNBvhBinaryDataOutputIsCompression  
(PNBOOL isCompression);

Parameters

*isCompression*

Compress or not. Set TRUE if do, otherwise set FALSE.

### PNEnableBvhDataGlobalDisplacement

Enable global or local coordinate of displacement that bvh data exporting.

PNLIB\_API void PNEnableBvhDataGlobalDisplacement(int avatarIndex,   
BOOL isGlobalDips);

Parameters

*avatarIndex*

Avatar index.

*isGlobalDips*

BOOL type variable. Set TRUE if is in global coordinate, otherwise set FALSE.

### PNRotateFaceDirection

Set rotation of face direction.

PNLIB\_API void PNRotateFaceDirection(int avatarIndex, float yaw);

Parameters

*avatarIndex*

Avatar index.

*yaw*

Float type variable means yawing degree of avatar.

### PNRotateModel

Set rotation of avatar pitching and rolling degree.

PNLIB\_API void PNRotateModel(int avatarIndex, float pitch,   
float roll);

Parameters

*avatarIndex*

Avatar index.

*pitch*

Float type variable means pitching degree of avatar.

*roll*

Float type variable means rolling degree of avatar.

### PNZeroOutAllAvatar

Zero out all avatar.

PNLIB\_API void PNZeroOutAllAvatar();

Remarks

In XY plane.

### PNZeroOutPosition

Zero out the avatar position.

PNLIB\_API void PNZeroOutPosition(int avatarIndex);

Parameters

*avatarIndex*

Avatar index.

Remarks

In XY plane.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Action Recognition \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### PNEnableActionRecognition

Enable action recognition.

PNLIB\_API void PNEnableActionRecognition(BOOL enable);

Parameters

*enable*

BOOL type variable. Set TRUE if enable action recognition, otherwise set FALSE.

Remarks

Default state is disabled.

# Error code

|  |  |  |
| --- | --- | --- |
| Code | Message | Remarks |
| 0 | No Error. |  |
| 1 | Init failed. |  |
| 2 | Load raw data failed. Too many avatar count. |  |
| 3 | Allocate memory failed. |  |
| 4 | Buffer is full. |  |
| 5 | Failed to start thread. |  |
| 6 | Calibration data file is not exist. |  |
| 7 | Open file failed. |  |
| 8 | Unknown rotateOrder. |  |
| 9 | Constact edit file format error. |  |
| 10 | Rewind file failed. |  |
| 11 | Loading file failed: No enough memory. |  |
| 12 | Fbx format check error. |  |
| 13 | Fbx format check error. |  |
| 14 | Raw data version: No version information. |  |
| 15 | Failed to enter time critical mode. |  |
| 16 | Not a calibration data file. |  |
| 17 | Unable to retrieve data descriptions. |  |
| 18 | Unknown data type. |  |
| 19 | Open fcd file failed. |  |
| 20 | Index is out of range. |  |
| 21 | Incorrect bone index. |  |
| 22 | You can not set to unknown type. |  |
| 23 | Function is not implemented. |  |
| 24 | 'AppDataFolder' or 'WorkingFolder' paramenter is illegal. |  |
| 25 | Buffer is too small. |  |
| 26 | Invalid pointer. |  |
| 27 | Can not be less than or equal to 0. |  |
| 28 | Warning: The defalut forlder of application directory will be used. |  |
| 29 | Value out of range. |  |
| 30 | Running mode is error. |  |
| 31 | Raw file is not exist or open raw file failed. |  |
|  |  |  |
| 100 | CreateBvhPlayer FAILED. |  |
| 101 | Calculate frame count error. |  |
| 102 | Is not a raw data file. |  |
| 103 | No data in raw data file. |  |
| 104 | Read data error. |  |
| 105 | Load raw data failed. Too many avatar count. |  |
| 106 | Play raw data failed. |  |
| 107 | Open raw file failed: wrong frame count. |  |
| 108 | Runing Mode failed. |  |
|  |  |  |
| 200 | Unable to connect to server. Host not present. |  |
| 201 | Unable to connect to server. /\*有错误码传入\*/ |  |
| 202 | error un-initting Client. |  |
| 103 | error re-initting Client. |  |
|  |  |  |
| 300 | The current sensor binding is not comfortable with the specified sensor combination mode. |  |
| 301 | Unsupported mode. |  |
| 302 | No sensor binded. |  |
| 303 | The sensor is not bound bone. |  |
|  |  |  |
| 400 | The bone index is out of range. |  |
| 401 | Bone length should be large than 0. |  |
| 402 | Bone index of BVH output sequence error. |  |
| 403 | The refered parent node is not exist in bone system. |  |
|  |  |  |
| 500 | Avatar index error. |  |
| 501 | No avatar to deal with this data. |  |
| 502 | The sensor is not beyond the avatar. |  |
| 503 | Avatar name is too long |  |
| 504 | Bone index error. |  |
|  |  |  |
| 600 | Bone index error. |  |
|  |  |  |
| 700 | Eable constraint edit failed: Constraint editing callback handle not set.\nPlease set handle befor enable constraint edit function. |  |
| 701 | Wrong frame index. |  |
| 702 | Wrong level specified. The max level is 10. |  |
| 703 | Start frameIndex should be less than end frameIndex. |  |
|  |  |  |
| 800 | This function is currently not supported in this version. |  |