Neuron Data Reader Runtime API

Documentation

By Yuanhui He
Yufeng Tang

27.01.2016

NeuronDataReader b16

Noitom Technology Co., Ltd.

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

Contents

1. Overview

1.1. NeuronDataReader framework

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

The structure of NeuronDataReader library is shown below.

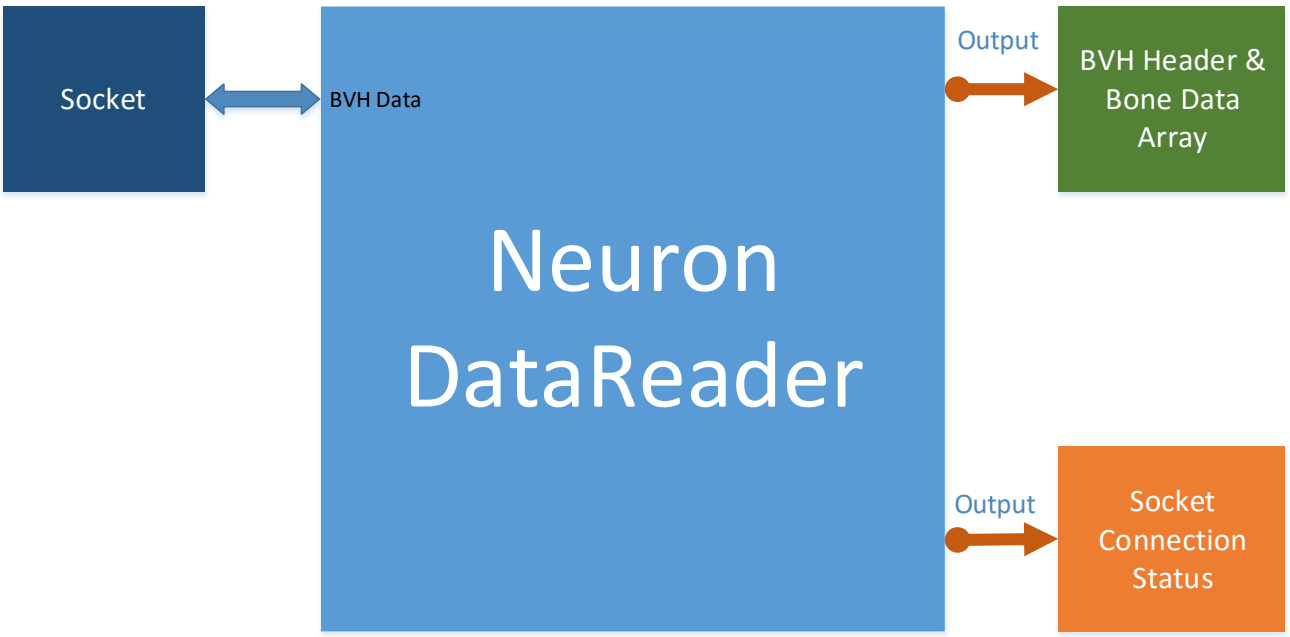


Fig. 1-1 NeuronDataReader Overview

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

1.2. Skeleton Data Format

In this version, skeleton data include BVH data and calculation data.

1.2.1. BVH Data

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

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

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

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

bones only have rotation data.

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

1) BVH data with displacement

Displacement_X = bone index * 6 + 0

Displacement_Y = bone index * 6 + 1

Displacement_Z = bone index * 6 + 2

Rotation_Y = bone index * 6 + 3

Rotation_X = bone index * 6 + 4

Rotation_Z = bone index * 6 + 5

2) BVH data without displacement

Except rotation, only the hip node has displacement data.

Root_Displacement_X = 0

Root_Displacement_Y = 1

Root_Displacement_Z = 2

Rotation_Y = 3 + bone index * 3 + 0

Rotation_X = 3 + bone index * 3 + 1

Rotation_Z = 3 + bone index * 3 + 2

3) BVH data with Reference

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

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

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

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

NOTE: the displacement of the BVH data is a constant usually, it reflects the offset of the initial position of the skeleton relative to the initial position of the parent node of the TPose. So it is a constant.

1.2.2. Calculation Data

Through the Network, NeuronDataReader receives calculation data frames from Axis Neuron, and calculation data in each frame include all the sensor data and motion data of 21 bones, and at last also include the contact state of two feet.

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

The contact judgment of right and left foot also is contained at the last of each frame data: 1 means contacting to the ground, 0 means not contact, with float type, 4bytes.

So, for a calculation data frame contains 21 * 16 + 2 = 338 (float), 338 * 4 = 1352(bytes).

The calculation data format is shown as below:
Position_X = bone index * 16 + 0
Position_Y = bone index * 16 + 1
Position_Z = bone index * 16 + 2

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

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

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

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

The output order in each frame data is No.1 to No.21. The bone index or output order for calculation data could be found in Appendix C and Appendix D.

1.3. Data Frequency

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

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

1.4. User Agreement

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

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

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

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

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

2. Reference

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

2.1. Data type definitions

2.1.1.Cross-platform data types

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

```
#ifdef __OS_XUN__           // Mac OS X, Linux or Unix like OS data type definition
#define __stdcall           // Empty
#endif

#ifndef NBOOL
#define NBOOL int
#endif

#ifndef TRUE
#define TRUE  1
#endif

#ifndef FALSE
#define FALSE 0
#endif
```

2.1.2.Socket connection status

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

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

2.1.3.Data version of stream

For different versions of NeuronDataReader, the data structure for communication could be changed, both in meaning and structure. Data version is used to be compatible with the data generated by old version of NeuronDataReader.

```
// Data version
```

```
typedef union 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;
```

2.1.4.Header of BVH data stream

```
// Header format of BVH data
typedef struct _BvhDataHeader
{
    uint16_t   Token1;          // Package start token: 0xDDFF
    DATA_VER   DataVersion;     // Version of community data format. e.g.: 1.0.0.2
    uint16_t   DataCount;       // Values count
    uint8_t    WithDisp;        // With/out displacement
    uint8_t    WithReference;   // With/out reference bone data at first
    uint32_t   AvatarIndex;     // Avatar index
    uint8_t    AvatarName[32];  // Avatar name
    uint32_t   FrameIndex;      // Frame data index
    uint32_t   Reserved;        // Reserved, only enable this package has 64bytes length
    uint32_t   Reserved1;       // Reserved, only enable this package has 64bytes length
    uint32_t   Reserved2;       // Reserved, only enable this package has 64bytes length
    uint16_t   Token2;          // Package end token: 0xEEFF
}BvhDataHeader;
```

2.1.5.Header of Calculation data stream

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

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

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

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

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

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

2.2. Callbacks and callback register

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

2.2.1.Skeleton data callback

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

Parameters

customedObj

User defined object.

sender

Connector reference of TCP/IP client as identity.

header

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

data

Float type array pointer, to output binary data.

Remarks

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

2.2.2.Calculation data callback

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

Parameters

customedObj

User defined object.

sender

Connector reference of TCP/IP client as identity.

Pack

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

data

Float type array pointer, to output binary data.

Remarks

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

2.2.3.Socket status callback

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

Parameters

customedObj

User defined object.

sender

Connector reference of TCP/IP client as identity.

status

Indicate the status changes of current socket.

message

Status description.

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

2.3. API reference

2.3.1.BRRegisterFrameDataCallback

Register the BVH data receiving callback handle:

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

Parameters

customedObj

User defined object.

handle

A function pointer of FrameDataReceived type.

Remarks

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

2.3.2.BRRegisterCalculationDataCallback

Register the calculation data receiving callback handle:

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

Parameters

customedObj

User defined object.

handle

A function pointer of CalculationDataReceived type.

Remarks

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

2.3.3.BRRegisterSocketStatusCallback

Register socket status callback Handle:

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

Parameters

customedObj

User defined object.

handle

A function pointer.

Remarks

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

2.3.4.BRConnectTo

Connect to the server with given IP address and port:

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

Parameters

serverIP

Server's IP address.

nPort

Server's port.

Return Values

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

2.3.5.BRStartUDPServiceAt

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

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

2.3.6.BRCloseSocket

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

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

2.3.7.BRGetSocketStatus

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

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

Return Values

Return the status of referred socket.

2.3.8.BRGetLastErrorMessage

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

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

Return Values

Return the last error message.

Remarks

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

3. Illustrations

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

3.1. C++ demo with MFC

Bellow will descript how library be used in C++.

For windows platform,

1. Start Visual Studio 2013, select “New Project…” in the start page.

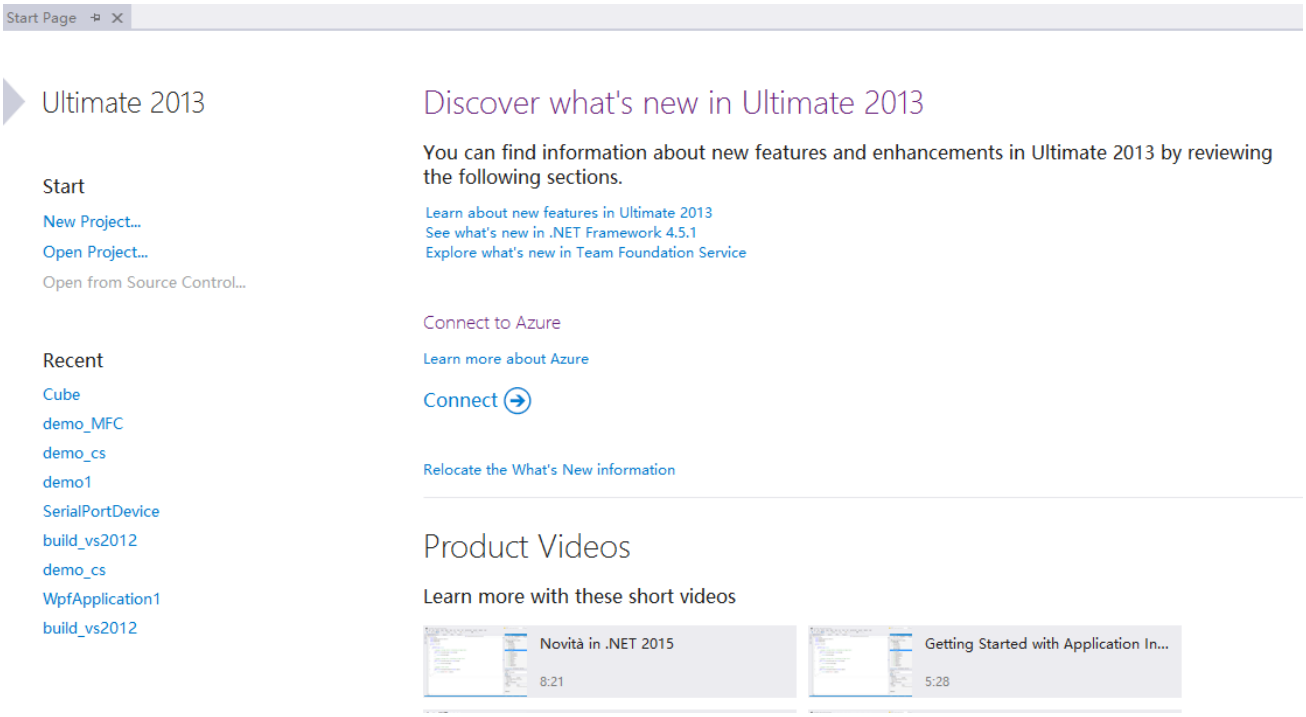


Fig. 3-1

2. Choose Visual C++--> MFC Application in the template list, name the project as “demo_MFC”.

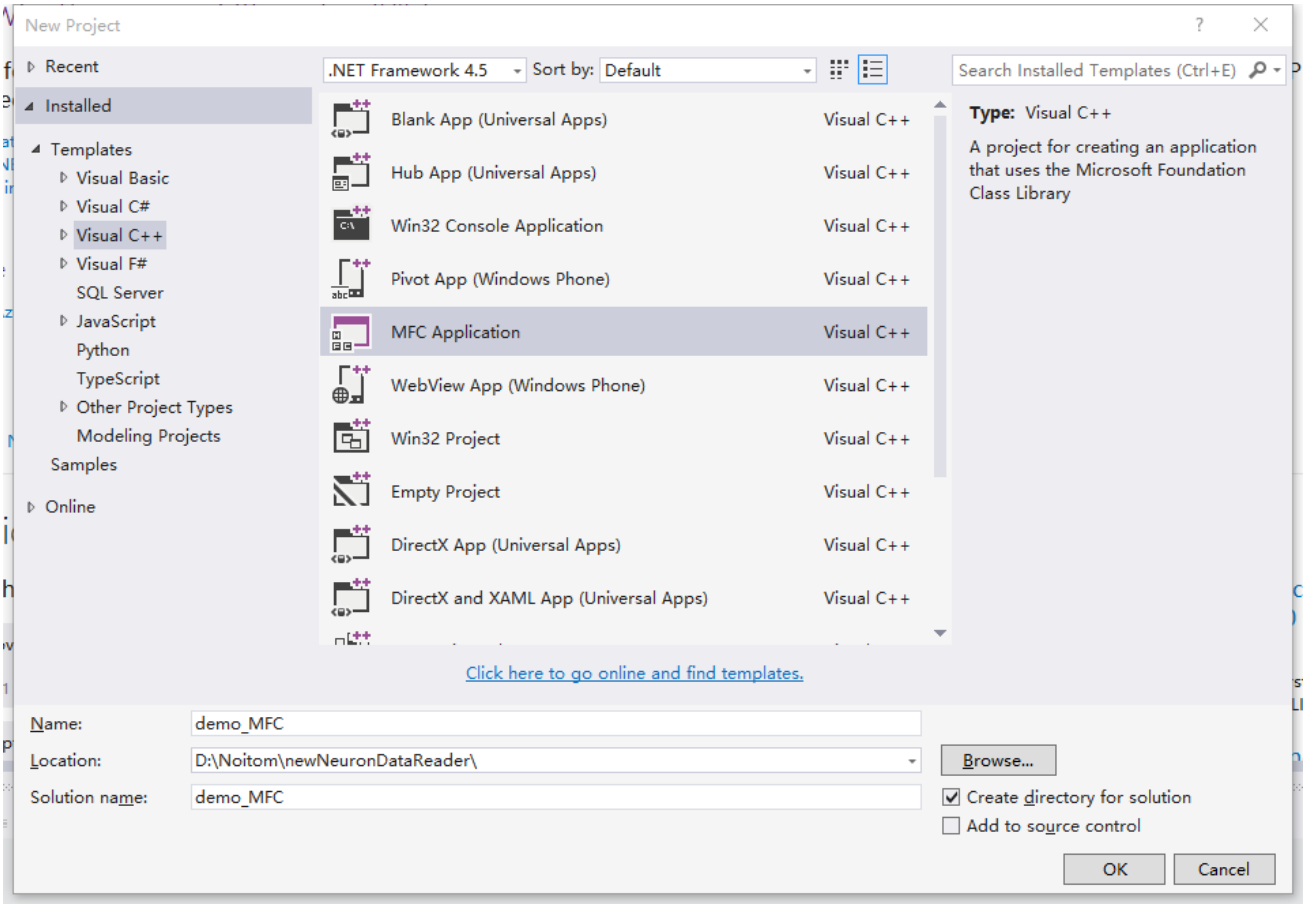


Fig. 3-2

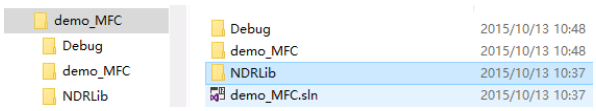
3. Click OK, and an empty MFC project is created.

4. Select dialog application type.

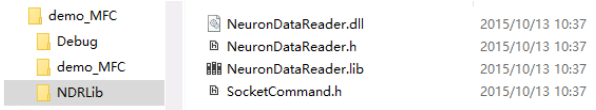
5. Finish the creation steps. A dialog based application is created.

In order to use the NeuronDataReader SDK, users must include the head file of NeuronDataReader, and then link the library of NeuronDataReader when compiling.

The NeuronDataReader SDK files is in the root of demo:



The files of NeuronDataReader:



The relevant codes are shown below.

demo_MFCDlg.h file:

```
// demo_MFCDlg.h : header file
//

#pragma once

// Include the NeuronDataReader head file
#include "../NDRLib/NeuronDataReader.h"

#include "afxwin.h"

#define WM_UPDATE_MESSAGE (WM_USER+200)

// Cdemo_MFCDlg dialog
class Cdemo_MFCDlg : public CDialogEx
{
// Construction
public:
	Cdemo_MFCDlg(CWnd* pParent = NULL);	// standard constructor

	enum
	{
		BVHBoneCount = 59,
		CalcBoneCount = 21,
	};
// Dialog Data
	enum { IDD = IDD_DEMO_MFC_DIALOG };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support

	static void __stdcall bvhFrameDataReceived(void* customedObj, SOCKET_REF sender, BvhDataHeaderEx* header,
float* data);

	// Receive calculation data
	static void __stdcall CalcFrameDataReceive( void* customObje, SOCKET_REF sender, CalcDataHeader* header,
float* data );
```

```
// Implementation
protected:
	HICON m_hIcon;

	SOCKET_REF sockTCPRef;
	SOCKET_REF sockUDPRef;

	void showBvhBoneInfo(SOCKET_REF sender, BvhDataHeaderEx* header, float* data);

	void showCalcBoneInfo(SOCKET_REF sender, CalcDataHeader* header, float* data );

	// Generated message map functions
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();

	afx_msg LRESULT OnUpdateMessage( WPARAM wParam, LPARAM lParam );
	DECLARE_MESSAGE_MAP()
public:
	CComboBox m_wndComBoxBone;

	CString m_strIPAddress;
	CString m_strPort;
	CString m_strUDPPort;

	afx_msg void OnBnClickedOk();
	afx_msg void OnBnClickedCancel();
	afx_msg void OnBnClickedButtonTcpConnection();
	afx_msg void OnBnClickedButtonUdpConnection();
	afx_msg void OnCbnSelchangeComboSelectionBoneIndex();
	CButton m_radioBvh;
	afx_msg void OnBnClickedRadio();

	void UpdateBvhDataShowUI();
	void UpdateCalcDataShowUI();
	CString m_clacAx;
	CString m_calcAy;
	CString m_calcAz;
	CString m_calcGx;
	CString m_calcGy;
	CString m_calcGz;
	CString m_calcPx;
	CString m_calcPy;
```

```
    CString m_calcQs;
    CString m_calcQx;
    CString m_calcQy;
    CString m_calcQz;
    CString m_calcVx;
    CString m_calcVy;
    CString m_calcVz;
    CString m_calcPz;
    CString m_bvhAngleZ;
};
```

demo_MFCDlg.cpp file:

```
// demo_MFCDlg.cpp : implementation file

#include "stdafx.h"
#include "demo_MFC.h"
#include "demo_MFCDlg.h"
#include "afxdialogex.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// Load SDK library
#pragma comment(lib, "../NDRLib/NeuronDataReader.lib")//Add Lib


// CAboutDlg dialog used for App About

class CAboutDlg : public CDialogEx
{
public:
	CAboutDlg();

// Dialog Data
	enum { IDD = IDD_ABOUTBOX };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support

// Implementation
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
```

```
END_MESSAGE_MAP()


// Cdemo_MFCDlg dialog

Cdemo_MFCDlg::Cdemo_MFCDlg(CWnd* pParent /*=NULL*/)
	: CDialogEx(Cdemo_MFCDlg::IDD, pParent)
	, m_strIPAddress(_T(""))
	, m_strPort(_T(""))
	, m_strUDPPort(_T(""))

	// Calc data
	, m_calcPx( _T( "" ) )
	, m_calcPy( _T( "" ) )
	, m_calcPz( _T( "" ) )

	, m_calcVx( _T( "" ) )
	, m_calcVy( _T( "" ) )
	, m_calcVz( _T( "" ) )

	, m_calcQs( _T( "" ) )
	, m_calcQx( _T( "" ) )
	, m_calcQy( _T( "" ) )
	, m_calcQz( _T( "" ) )

	, m_clacAx( _T( "" ) )
	, m_calcAy( _T( "" ) )
	, m_calcAz( _T( "" ) )

	, m_calcGx( _T( "" ) )
	, m_calcGy( _T( "" ) )
	, m_calcGz( _T( "" ) )

	, m_bvhAngleZ( _T( "" ) )
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);

	sockTCPRef = NULL;
	sockUDPRef = NULL;

}

void Cdemo_MFCDlg::DoDataExchange(CDataExchange* pDX)
{
```

```
	CDialogEx::DoDataExchange( pDX );
	DDX_Text( pDX, IDC_TEXT1, m_strIPAddress );
	DDX_Text( pDX, IDC_TEXT2, m_strPort );
	DDX_Text( pDX, IDC_UDP, m_strUDPPort );
	DDX_Control( pDX, IDC_RADIO_BVH, m_radioBvh );
	DDX_Text( pDX, IDC_STATIC_AX, m_clacAx );
	DDX_Text( pDX, IDC_STATIC_AY, m_calcAy );
	DDX_Text( pDX, IDC_STATIC_AZ, m_calcAz );
	DDX_Text( pDX, IDC_STATIC_GX, m_calcGx );
	DDX_Text( pDX, IDC_STATIC_GY, m_calcGy );
	DDX_Text( pDX, IDC_STATIC_GZ, m_calcGz );
	DDX_Text( pDX, IDC_STATIC_PX, m_calcPx );
	DDX_Text( pDX, IDC_STATIC_PY, m_calcPy );
	DDX_Text( pDX, IDC_STATIC_QS, m_calcQs );
	DDX_Text( pDX, IDC_STATIC_QX, m_calcQx );
	DDX_Text( pDX, IDC_STATIC_QY, m_calcQy );
	DDX_Text( pDX, IDC_STATIC_QZ, m_calcQz );
	DDX_Text( pDX, IDC_STATIC_VX, m_calcVx );
	DDX_Text( pDX, IDC_STATIC_VY, m_calcVy );
	DDX_Text( pDX, IDC_STATIC_VZ, m_calcVz );
	DDX_Text( pDX, IDC_STATIC_PZ, m_calcPz );
	DDX_Control( pDX, IDC_COMBO_SELECTION_BONE_INDEX, m_wndComBoxBone );
	DDX_Text( pDX, IDC_STATIC_ANGLE_Z, m_bvhAngleZ );
}

BEGIN_MESSAGE_MAP(Cdemo_MFCDlg, CDialogEx)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDOK, &Cdemo_MFCDlg::OnBnClickedOk)
	ON_BN_CLICKED(IDCANCEL, &Cdemo_MFCDlg::OnBnClickedCancel)
	ON_BN_CLICKED(IDC_BUTTON_TCP_CONNECTION, &Cdemo_MFCDlg::OnBnClickedButtonTcpConnection)
	ON_BN_CLICKED(IDC_BUTTON_UDP_CONNECTION, &Cdemo_MFCDlg::OnBnClickedButtonUdpConnection)
	ON_CBN_SELCHANGE(IDC_COMBO_SELECTION_BONE_INDEX,
&Cdemo_MFCDlg::OnCbnSelchangeComboSelectionBoneIndex)
	ON_BN_CLICKED( IDC_RADIO_BVH, &Cdemo_MFCDlg::OnBnClickedRadio )
	ON_BN_CLICKED( IDC_RADIO_CALC, &Cdemo_MFCDlg::OnBnClickedRadio )
	ON_MESSAGE( WM_UPDATE_MESSAGE, OnUpdateMessage )
END_MESSAGE_MAP()


// Cdemo_MFCDlg message handlers

BOOL Cdemo_MFCDlg::OnInitDialog()
```

```
{
	CDialogEx::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		BOOL bNameValid;
		CString strAboutMenu;
		bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
		ASSERT(bNameValid);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here
	BRRegisterFrameDataCallback(this, bvhFrameDataReceived);

	BRRegisterCalculationDataCallback(this, CalcFrameDataReceive);

	m_strIPAddress = L"127.0.0.1";//Default IP Address
	m_strPort = L"7001";//Default Port

	m_strUDPPort = L"7001"; // Default UDP Port

	m_radioBvh.SetCheck( BST_CHECKED );
	UpdateBvhDataShowUI();
	UpdateData(FALSE);

	return TRUE;  // return TRUE  unless you set the focus to a control
}
```

```
void __stdcall Cdemo_MFCDlg::bvhFrameDataReceived(void* customedObj, SOCKET_REF sender, BvhDataHeaderEx*
header, float* data)
{
	Cdemo_MFCDlg* pthis = (Cdemo_MFCDlg*)customedObj;
	pthis->showBvhBoneInfo(sender, header, data);
}

void __stdcall Cdemo_MFCDlg::CalcFrameDataReceive( void* customObje, SOCKET_REF sender, CalcDataHeader*
header, float* data )
{
	Cdemo_MFCDlg* pthis = (Cdemo_MFCDlg*)customObje;
	pthis->showCalcBoneInfo( sender, header, data );

}

void Cdemo_MFCDlg::showBvhBoneInfo(SOCKET_REF sender, BvhDataHeaderEx* header, float* data)
{
	USES_CONVERSION;

	// show frame index
	char strFrameIndex[60];
	itoa(header->FrameIndex, strFrameIndex, 10);
	GetDlgItem(IDC_STATIC_FRAME_INDEX)->SetWindowText(A2W(strFrameIndex));

	// calculate data index for selected bone
	int dataIndex = 0;
	int curSel = m_wndComBoxBone.GetCurSel();
	if ( curSel == CB_ERR ) return;

	if (header->WithDisp)
	{
		dataIndex = curSel * 6;
		if (header->WithReference)
		{
			dataIndex += 6;
		}

		float dispX = data[dataIndex + 0];
		float dispY = data[dataIndex + 1];
		float dispZ = data[dataIndex + 2];

		char strBuff[32];
		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", dispX);
```

```
		GetDlgItem(IDC_STATIC_DISP_X)->SetWindowText(A2W(strBuff));

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", dispY);
		GetDlgItem(IDC_STATIC_DISP_Y)->SetWindowText(A2W(strBuff));

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", dispZ);
		GetDlgItem(IDC_STATIC_DISP_Z)->SetWindowText(A2W(strBuff));

		float angX = data[dataIndex + 4];
		float angY = data[dataIndex + 3];
		float angZ = data[dataIndex + 5];

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angX);
		GetDlgItem(IDC_STATIC_ANGLE_X)->SetWindowText(A2W(strBuff));

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angY);
		GetDlgItem(IDC_STATIC_ANGLE_Y)->SetWindowText(A2W(strBuff));

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angZ);
		GetDlgItem(IDC_STATIC_ANGLE_Z)->SetWindowText(A2W(strBuff));

	}
	else
	{
		if (curSel == 0)
		{
			dataIndex = 0;
			if (header->WithReference)
			{
				dataIndex += 6;
			}

			// show hip's displacement
			float dispX = data[dataIndex + 0];
			float dispY = data[dataIndex + 1];
			float dispZ = data[dataIndex + 2];

			char strBuff[32];
			sprintf_s(strBuff, sizeof(strBuff), "%0.3f", dispX);
			GetDlgItem(IDC_STATIC_DISP_X)->SetWindowText(A2W(strBuff));

			sprintf_s(strBuff, sizeof(strBuff), "%0.3f", dispY);
			GetDlgItem(IDC_STATIC_DISP_Y)->SetWindowText(A2W(strBuff));
```

```
		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", dispZ);
		GetDlgItem(IDC_STATIC_DISP_Z)->SetWindowText(A2W(strBuff));

		// show hip's angle
		float angX = data[dataIndex + 4];
		float angY = data[dataIndex + 3];
		float angZ = data[dataIndex + 5];

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angX);
		GetDlgItem(IDC_STATIC_ANGLE_X)->SetWindowText(A2W(strBuff));

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angY);
		GetDlgItem(IDC_STATIC_ANGLE_Y)->SetWindowText(A2W(strBuff));

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angZ);
		GetDlgItem(IDC_STATIC_ANGLE_Z)->SetWindowText(A2W(strBuff));
	}
	else
	{
		dataIndex = 3 + curSel * 3;
		if (header->WithReference)
		{
			dataIndex += 6;
		}

		// show displacement
		char strBuff[32];
		sprintf_s(strBuff, sizeof(strBuff), "NaN");
		GetDlgItem(IDC_STATIC_DISP_X)->SetWindowText(A2W(strBuff));
		GetDlgItem(IDC_STATIC_DISP_Y)->SetWindowText(A2W(strBuff));
		GetDlgItem(IDC_STATIC_DISP_Z)->SetWindowText(A2W(strBuff));

		// show angle
		float angX = data[dataIndex + 1];
		float angY = data[dataIndex + 0];
		float angZ = data[dataIndex + 2];

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angX);
		GetDlgItem(IDC_STATIC_ANGLE_X)->SetWindowText(A2W(strBuff));

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angY);
		GetDlgItem(IDC_STATIC_ANGLE_Y)->SetWindowText(A2W(strBuff));

		sprintf_s(strBuff, sizeof(strBuff), "%0.3f", angZ);
```

```
                GetDlgItem(IDC_STATIC_ANGLE_Z)->SetWindowText(A2W(strBuff));
            }
        }
}

LRESULT Cdemo_MFCDlg::OnUpdateMessage( WPARAM wParam, LPARAM lParam )
{
    UpdateData( FALSE );
    return 0;
}

void Cdemo_MFCDlg::showCalcBoneInfo( SOCKET_REF sender, CalcDataHeader* header, float* data )
{
    USES_CONVERSION;
    // show frame index
    char strFrameIndex[60];
    itoa( header->FrameIndex, strFrameIndex, 10 );
    GetDlgItem( IDC_STATIC_FRAME_INDEX )->SetWindowText( A2W( strFrameIndex ) );

    int dataIndex = 0;
    int curSel = m_wndComBoxBone.GetCurSel();
    if ( curSel == CB_ERR ) return;

    if ( curSel > CalcBoneCount ) return;

    dataIndex = 16 * curSel;
    //CString tmptmp;
    //tmptmp.Format( L"%d\n", dataIndex );
    //OutputDebugString( tmptmp );

    CString tmpData;
    tmpData.Format( L"%0.3f", data[dataIndex + 0] );
    m_calcPx = tmpData;
    tmpData.Format( L"%0.3f", data[dataIndex + 1] );
    m_calcPy = tmpData;
    tmpData.Format( L"%0.3f", data[dataIndex + 2] );
    m_calcPz = tmpData;
    tmpData.Format( L"%0.3f", data[dataIndex + 3] );
    m_calcVx = tmpData;
    tmpData.Format( L"%0.3f", data[dataIndex + 4] );
    m_calcVy = tmpData;
    tmpData.Format( L"%0.3f", data[dataIndex + 5] );
    m_calcVz = tmpData;
    tmpData.Format( L"%0.3f", data[dataIndex + 6] );
```

```
	m_calcQs = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 7] );
	m_calcQx = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 8] );
	m_calcQy = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 9] );
	m_calcQz = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 10] );
	m_clacAx = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 11] );
	m_calcAy = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 12] );
	m_calcAz = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 13] );
	m_calcGx = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 14] );
	m_calcGy = tmpData;
	tmpData.Format( L"%0.3f", data[dataIndex + 15] );
	m_calcGz = tmpData;

	PostMessage(WM_UPDATE_MESSAGE,0,0);//OK  —  UpdateDate

}

void Cdemo_MFCDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialogEx::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void Cdemo_MFCDlg::OnPaint()
{
	if (IsIconic())
```

```
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialogEx::OnPaint();
	}
}

// The system calls this function to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR Cdemo_MFCDlg::OnQueryDragIcon()
{
	return static_cast<HCURSOR>(m_hIcon);
}

void Cdemo_MFCDlg::OnBnClickedOk()
{
	//CDialogEx::OnOK();

}

void Cdemo_MFCDlg::OnBnClickedCancel()
{
	CDialogEx::OnCancel();
}

void Cdemo_MFCDlg::OnBnClickedButtonTcpConnection()
{
	UpdateData();
```

```
    if (sockTCPRef)
    {
        // close socket
        BRCloseSocket(sockTCPRef);

        // reconnect
        sockTCPRef = NULL;

        // change the title of button
        GetDlgItem(IDC_BUTTON_TCP_CONNECTION)->SetWindowText(L"Connect");
    }
    else
    {
        USES_CONVERSION;

        // get port number
        char* port = W2A(m_strPort);
        long nPort = 0;
        try
        {
            nPort = atoi(port);
        }
        catch (CException* e)
        {
            AfxMessageBox(L"Port number error", MB_OK);
            return;
        }

        // connect to remote server
        sockTCPRef = BRConnectTo(W2A(m_strIPAddress), nPort);

        // if success, change the title of button
        if (sockTCPRef)
        {
            GetDlgItem(IDC_BUTTON_TCP_CONNECTION)->SetWindowText(L"Disconnect");
        }
        else
        {
            // if failed, show message
            AfxMessageBox(A2W(BRGetLastErrorMessage()), MB_OK);
        }
    }
}
```

```
void Cdemo_MFCDlg::OnBnClickedButtonUdpConnection()
{
	UpdateData();

	if (sockUDPRef)
	{
		// close socket
		BRCloseSocket(sockUDPRef);

		// reconnect
		sockUDPRef = NULL;

		// change the title of button
		GetDlgItem(IDC_BUTTON_UDP_CONNECTION)->SetWindowText(L"Connect");
	}
	else
	{
		USES_CONVERSION;

		// get port number
		char* UDPport = W2A(m_strUDPPort);
		long nUDPPort = 0;
		try
		{
			nUDPPort = atoi(UDPport);
		}
		catch (CException* e)
		{
			AfxMessageBox(L"UDPPort number error", MB_OK);
			return;
		}

		// connect to remote server
		sockUDPRef = BRStartUDPServiceAt(nUDPPort);

		// if success, change the title of button
		if (sockUDPRef)
		{
			GetDlgItem(IDC_BUTTON_UDP_CONNECTION)->SetWindowText(L"Disconnect");
		}
		else
		{
			// if failed, show message
```

```
                AfxMessageBox(A2W(BRGetLastErrorMessage()), MB_OK);
            }
        }
}


void Cdemo_MFCDlg::OnCbnSelchangeComboSelectionBoneIndex()
{
    UpdateData();
}


void Cdemo_MFCDlg::OnBnClickedRadio()
{
    if (m_radioBvh.GetCheck() == BST_CHECKED)
    {
        UpdateBvhDataShowUI();
    }
    else
    {
        UpdateCalcDataShowUI();
    }
}

void Cdemo_MFCDlg::UpdateBvhDataShowUI()
{
    CString BoneID;
    m_wndComBoxBone.ResetContent();
    for ( int i = 0; i < BVHBoneCount; i++ )
    {
        BoneID.Format( L"%s%d", L"Bone", i );
        m_wndComBoxBone.AddString( BoneID );
    }
    m_wndComBoxBone.SetCurSel( 0 );
}

void Cdemo_MFCDlg::UpdateCalcDataShowUI()
{
    CString BoneID;
    m_wndComBoxBone.ResetContent();
    for ( int i = 0; i < CalcBoneCount; i++ )
    {
        BoneID.Format( L"%s%d", L"Bone", i );
        m_wndComBoxBone.AddString( BoneID );
```

```
        }
        m_wndComBoxBone.SetCurSel( 0 );
}
```

3.2. C Sharp demo

Bellow will illustrations how library be used in C#.

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

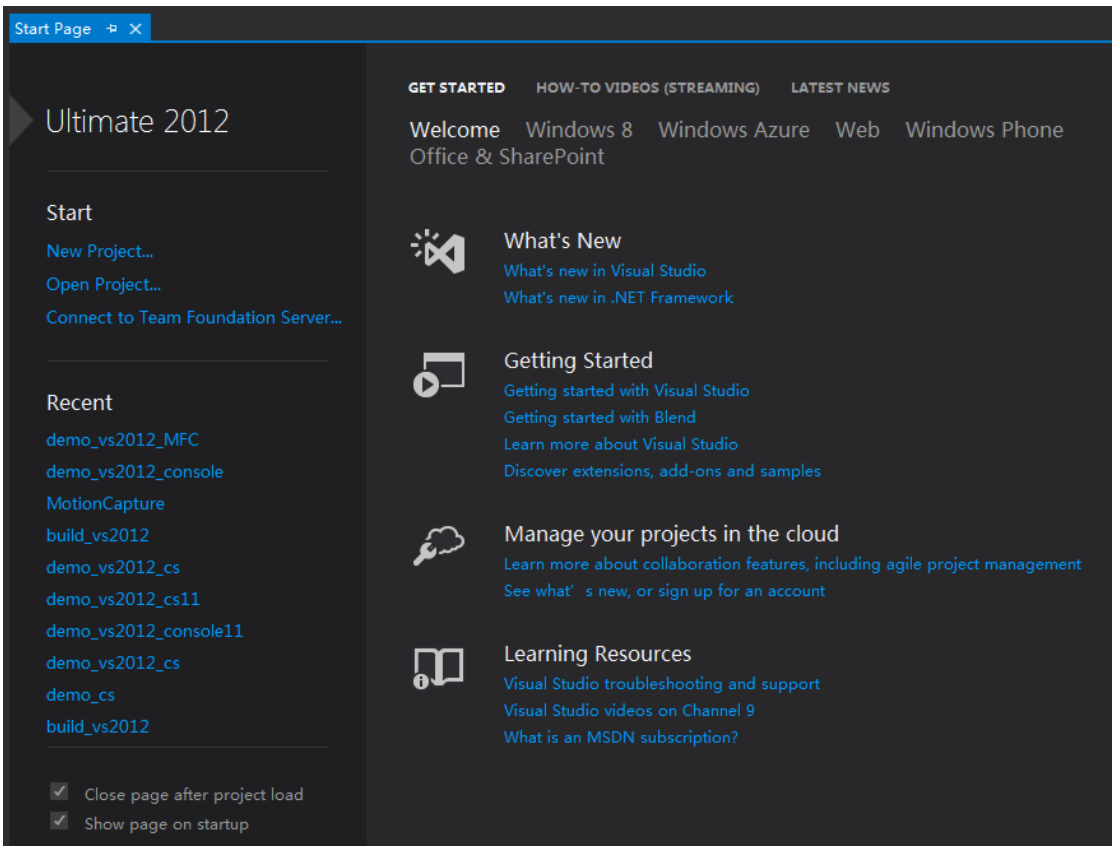


Fig. 3-3

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

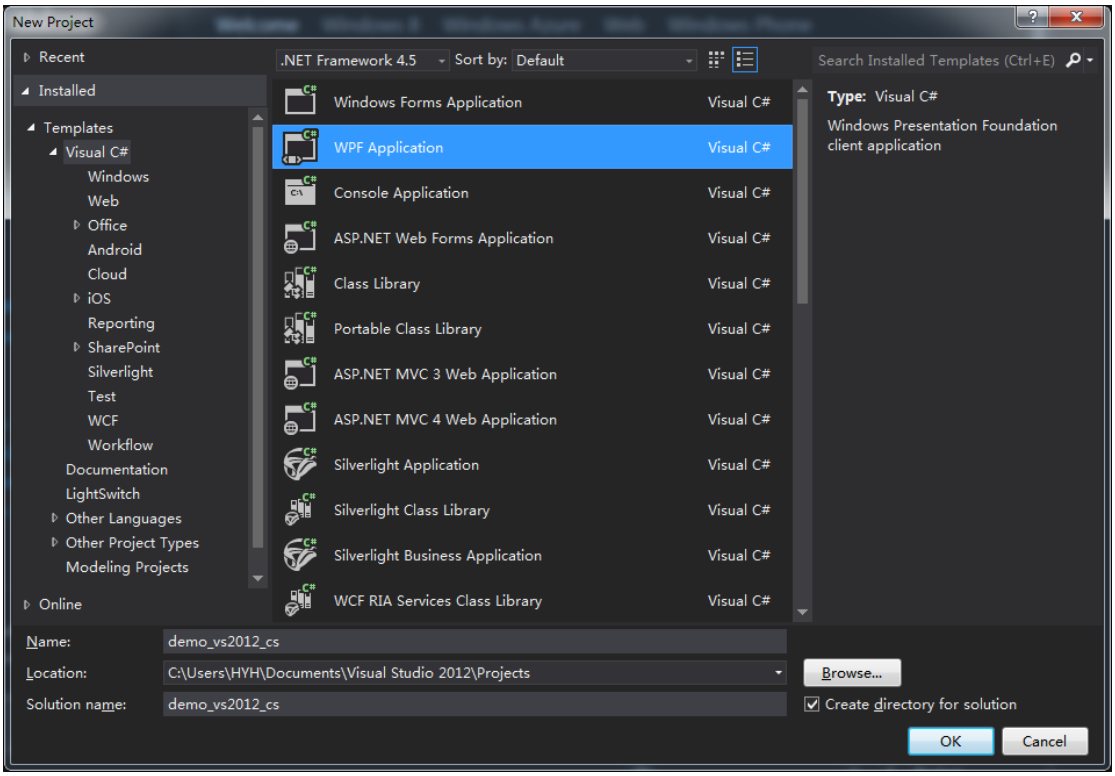


Fig. 3-4

3. Click OK, and an empty WPF project is created.

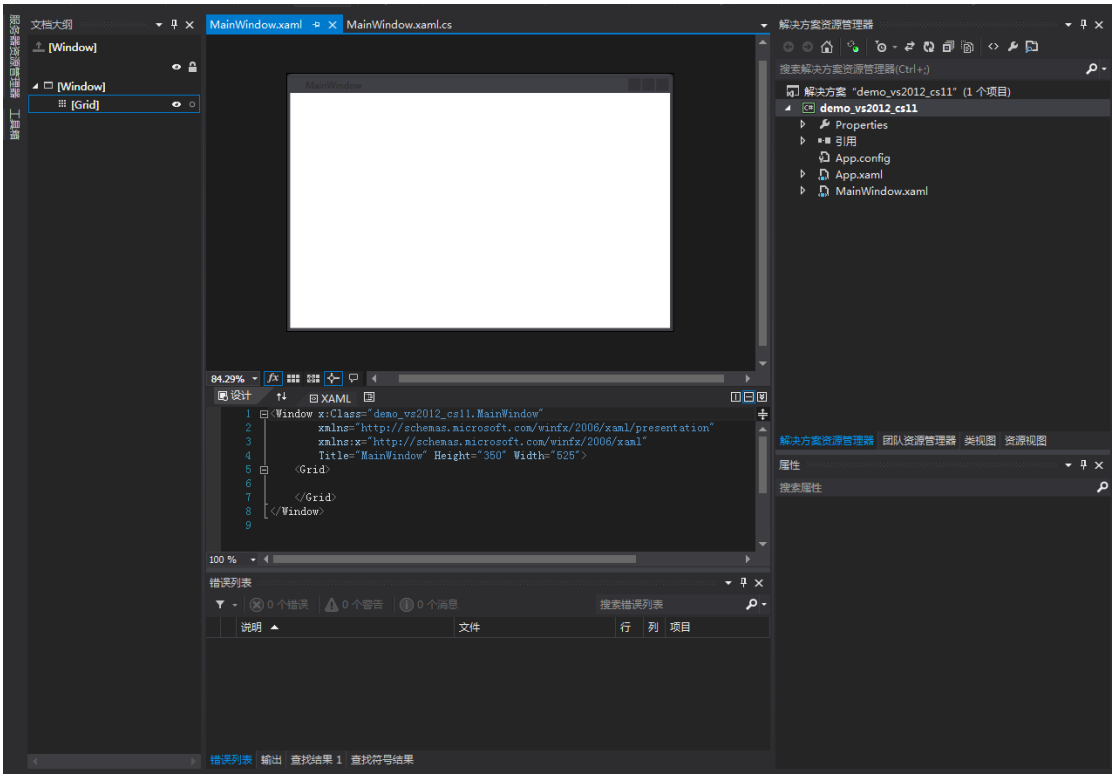


Fig. 3-5 WPF initial project

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

NeuronDataReader wrap class

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


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


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

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

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

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

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

```
    };

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

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


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

    #endregion
```

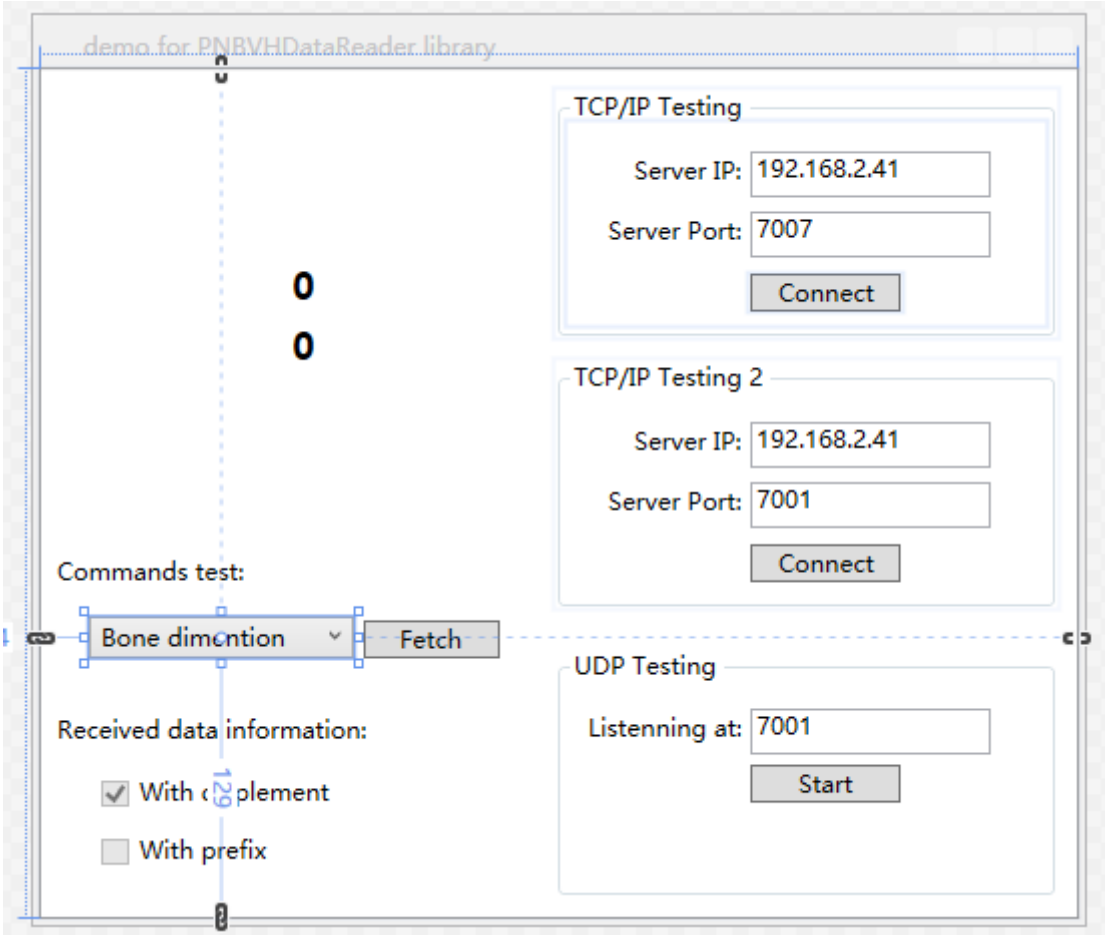
```
    #region Callbacks for data output
    /// <summary>
    /// FrameDataReceived CALLBACK
    /// Remarks
    /// The related information of the data stream can be obtained from BvhDataHeader.
    /// </summary>
    /// <param name="customObject">User defined object.</param>
    /// <param name="sockRef">Connector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

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

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

```
    #endregion

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

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

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

```
        }

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

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

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

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

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

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

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

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

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

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

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



Fir. 3-6 Interface configuration

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

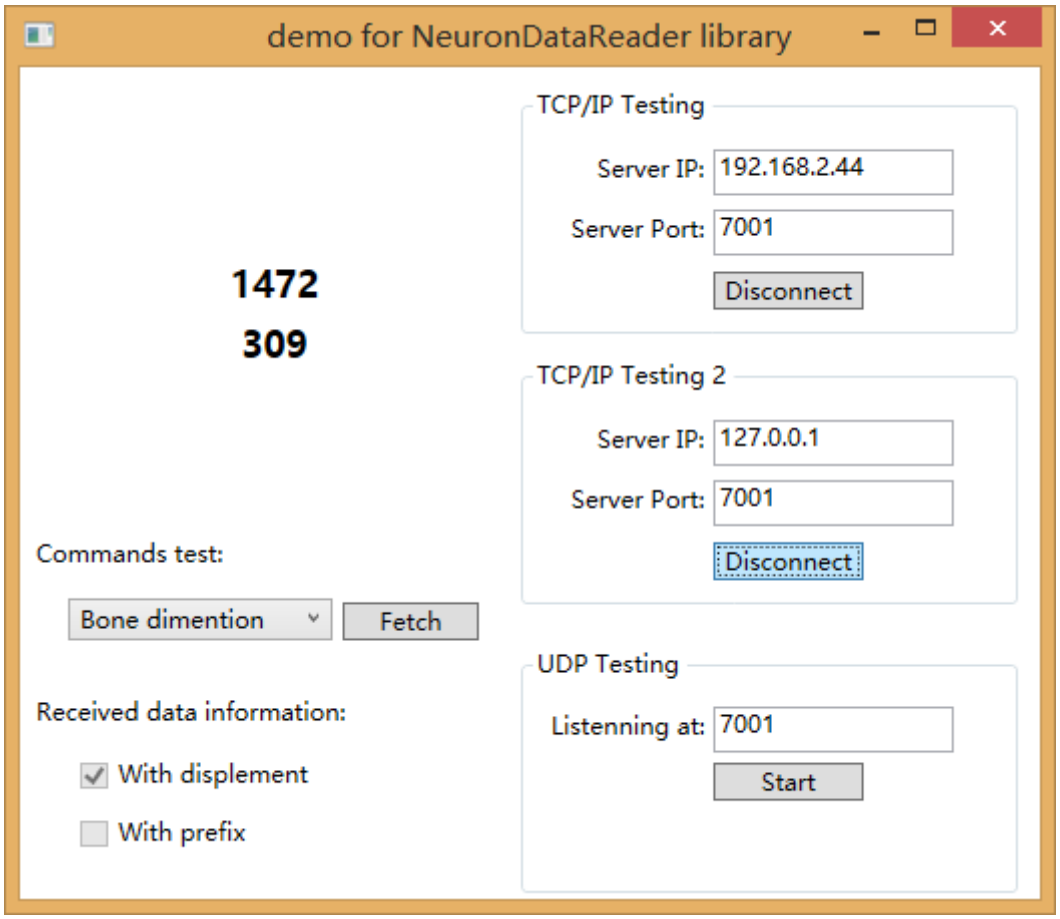


Fig. 3-7 Running Result

4. Known Bugs

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

Thank you!

Appendix A: Skeleton Data Sequence in Array

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

Appendix B: BVH Header Template

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

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

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

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

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

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

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

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

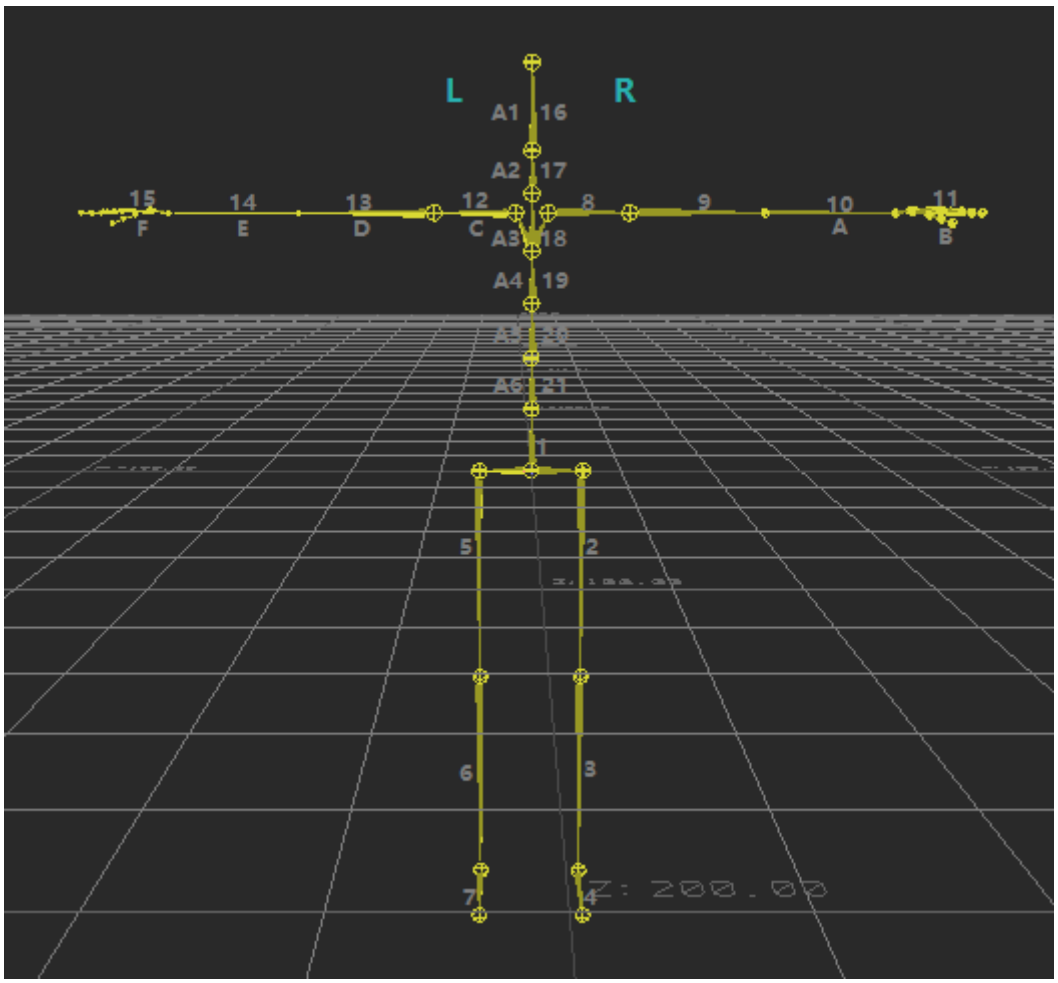
Frames: 0

Frame Time: 0.010

Appendix C: Bone Sequence Table

0. Hips
1. RightUpLeg
2. RightLeg
3. RightFoot
4. LeftUpLeg
5. LeftLeg
6. LeftFoot
7. RightShoulder
8. RightArm
9. RightForeArm
10. RightHand
11. LeftShoulder
12. LeftArm
13. LeftForeArm
14. LeftHand
15. Head
16. Neck
17. Spine3
18. Spine2
19. Spine1
20. Spine

Appendix D: Skeleton Graph



Revision history

Revision	Author	date	Description/changes
D1	Yuanhui He	12/22/2014	Initial released
D2	Peng Gao	12/22/2014	**Added:** Description of APIs. **Modified:** Format edit.
D3	Siyuan Deng	12/23/2014	**Added:** English translation. **Modified:**
D4	Yuanhui He	12/25/2014	**Added:** **Modified:** Delete string data stream type.
D5	Jinzhou Chen	12/25/2014	**Modified:** Modify the English translation.
D6	Yuanhui He	12/25/2014	**Modified:** Modify the English translation and some API description.
D7	Yuanhui He Siyuan Deng	1/26/2015	**Added:** Appendix, Skeleton Data format **Modified:** Data format description
D8	Yuanhui He	2/3/2015	**Added:** UDP protocol type support.
D9	Tobi	11/3/2015	**Modify:** English review.
D10	Yuanhui He	20/3/2015	**Add:** Multi-client is supported. **Modify:** English review.
D11	Yuanhui He	24/4/2015	**Add:** Some commands or APIs added to be used to sync parameters or data from server. **Delete:** Unity demo
D12	Yuanhui He	5/5/2015	**Modify:** Merged some TCP/UDP functions.
D13	Yufeng Tang	10/10/2015	**Add:** Details of skeleton data and data acquisition frequency description. C++ demo.

D14	Yufeng Tang	23/10/2015	**Add:** Details of calculation data and data acquisition frequency description. Appendix D: Bone index for calculation data.
D15	Yuanhui He	12/11/2015	**Add:** Appendix C: Bone Sequence Table
D16	Yufeng Tang	30/12/2015	**Add:** Chinese translation. **Modify:** Appendix A: Skeleton Data Sequence in Array. **Delete:** 2 command communication API in Section 2.3.