NOLO VR Android SDK Interfaces Documentation

LYRobotix Co., Ltd

May 2017

Directory

1. Introduction 1
2. SDK Interfaces Description 1
    2.1 Interfaces Detail 1
    2.2 Calling Process 4
Appendix: Special return value types in the interfaces 6

1. Introduction

NOLO VR Android SDK is the interfaces description which is provided by LYRobotix used for NOLO CV1, It is convenient for the APP accessor to integrate the SDK to get NOLO device data.

2. SDK Interfaces Description

2.1 Interfaces Detail

NOLO VR Android SDK has 13 interfaces, The name，function，functionality，parameter and return value of each interfaces are as follows.

Name	Description	
Interface of getting SDK instance	Function	UsbCustomTransfer getInstance(Context)
	Functionality	Get UsbCustomTransfer instance object
	Parameter	Context
	Return value	UsbCustomTransfer instance object
Interface of SDK initialization	Function	void usb_init()
	Functionality	Initialize the SDK
	Parameter	
	Return value	
Interface of SDK connecting to NOLO device	Function	int usb_conn()
	Functionality	Connect NOLO device interface
	Parameter	
	Return value	Return connection status code, 0: connection failed; 1: connection is successful

Interface of SDK sending data to NOLO device	Function	void usb_sendData(byte[] mbyte)
	Functionality	Send data to NOLO device
	Parameter	mbyte: data to be sent Byte[4]: [0xAA(First word of frame head), 0x66(Second word of frame head), 0x00(leftcontroller vibration intensity, in the range (0x00 ~ 0x64)), 0x00(rightcontroller vibration intensity, in the range (0x00 ~ 0x64))]
	Return value	
Interface of setting receive disconnection notification	Function	void setDisconnectedCallback(DisconnectedCallback mdis)
	Functionality	Interface of setting NOLO device disconnection notification, See the specific implementation of DisconnectedCallback in Demo;
	Parameter	mdis: APP receive the object of SDK disconnection notification
	Return value	
Interface of SDK to disconnect NOLO device	Function	void usb_finish()
	Functionality	Disconnect with NOLO device
	Parameter	
	Return value	
Interface of getting NOLO device version	Function	int getVersionByDeviceType(int type)
	Functionality	Get NOLO device version
	Parameter	Parameter type means device type, 0: headset; 1: leftcontroller; 2: rightcontroller;

		3:base station;
	Return value	Device version return value, 1: DK2; 2: CV1
Interface of getting NOLO device electricity quantity	Function	int getElectricityByDeviceType(int type)
	Functionality	Get NOLO device electricity quantity
	Parameter	Parameter type means device type， 0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station;
	Return value	NOLO device electricity quantity
Interface of getting NOLO device connection status	Function	int getDeviceTrackingStatus(int type)
	Functionality	Get NOLO device connection status
	Parameter	Parameter type means device type， 0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station;
	Return value	NOLO device connection status: 0: not connected or blocked; 1: normal
Interface of getting NOLO device position and attitude	Function	Nolo_Pose getPoseByDeviceType(int type)
	Functionality	Get NOLO device position and attitude information
	Parameter	Parameter type means device type， 0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station;
	Return value	Position and attitude information of NOLO device, see the attributes of Nolo_Pose in Demo
Interface of getting NOLO device feedback	Function	Nolo_ControllerStates getControllerStatesByDeviceType(int type)
	Functionality	Get NOLO device feedback information
	Parameter	Parameter type means device type，

		0: headset; 1: leftcontroller; 2: rightcontroller; 3:base station;
	Return value	Feedback information of NOLO device, see the attributes of Nolo_ControllerStates in Demo
Interface of getting NOLO device headset initial position	Function	Nolo_Vector3 getHmdInitPosition()
	Functionality	Get the coordinate point on surface when the helmet is calibrated
	Parameter	
	Return value	Return the coordinate point on surface when the helmet is calibrated
Interface of getting NOLO device headset calibration value	Function	int getHmdCalibration()
	Functionality	Get the calibration value between two points (This interface is valid only for the DK2 protocol of NOLO device)
	Parameter	
	Return value	The calibration value between two points

2.2 Calling Process

APP uses the UsbCustomTransfer class instance to complete all the interaction with the SDK request. The operation process is as follows.

Call UsbCustomTransfer.getInstance(Context mcontext) to get the UsbCustomTransfer object instance.

Call instance.usb_init() to initialize

Call instance.usb_conn() for usb device connection, Returns 1 if successful, 0 if it fails. Check if the device is inserted when it fails. If successful, you can operate the function interface which communicates to NOLO device, or wait for the device connection is successful

NOLO VR Android SDK provides an interface to the NOLO device communication function

When the APP exits, it is necessary to actively call the instance.usb_finish() interface to release the connection with the NOLO device before exiting

Appendix: Special return value types in the interfaces

```
public class Nolo_Vector3
{
    private float x;
    private float y;
    private float z;

    public void setX(float mx)
    {
        this.x = mx;
    }

    public void setY(float my) {
        this.y = my;
    }

    public void setZ(float mz) {
        this.z = mz;
    }

    public float getX() {
        return this.x;
    }

    public float getY() {
        return this.y;
```

```
    }

    public float getZ() {
        return this.z;
    }
}
public class Nolo_Quaternion
{
    private float x;
    private float y;
    private float z;
    private float w;

    public void setX(float mx)
    {
        this.x = mx;
    }

    public void setY(float my) {
        this.y = my;
    }

    public void setZ(float mz) {
        this.z = mz;
    }

    public void setW(float mw) {
        this.w = mw;
    }
```

```
    public float getX() {
        return this.x;
    }

    public float getY() {
        return this.y;
    }

    public float getZ() {
        return this.z;
    }

    public float getW() {
        return this.w;
    }
}
public class Nolo_Pose
{
    private Nolo_Vector3 pos;
    private Nolo_Quaternion rot;

    public void setPos(Nolo_Vector3 mpos)
    {
        this.pos = mpos;
    }

    public void setNolo_Quaternion(Nolo_Quaternion mrot) {
        this.rot = mrot;
```

```
    }

    public Nolo_Vector3 getPos() {
        return this.pos;
    }

    public Nolo_Quaternion getNolo_Quaternion() {
        return this.rot;
    }
}
public class Nolo_Vector2
{
    private float x;
    private float y;

    public void setX(float mx)
    {
        this.x = mx;
    }

    public void setY(float my) {
        this.y = my;
    }

    public float getX() {
        return this.x;
    }

    public float getY() {
```

```
        return this.y;
    }
}
public class Nolo_ControllerStates
{
    private int buttons;
    private int touches;
    private Nolo_Vector2 touchpadAxis;

    public void setButtons(int mbuttons)
    {
        this.buttons = mbuttons;
    }

    public void setTouches(int mtouches) {
        this.touches = mtouches;
    }

    public void setTouchpadAxis(Nolo_Vector2 mtouchpadAxis) {
        this.touchpadAxis = mtouchpadAxis;
    }

    public int getButtons() {
        return this.buttons;
    }

    public int getTouches() {
        return this.touches;
    }
```

```
    public Nolo_Vector2 getTouchpadAxis() {
        return this.touchpadAxis;
    }
}
public enum NoloButtonID
{
    TouchPad = 0,
    Trigger,
    Menu,
    System,
    Grip
}
public enum NoloTouchID
{
    TouchPad = 0
}
public class ButtonMask
{
    public const uint TouchPad = 1 << (int)NoloButtonID.TouchPad;
    public const uint Trigger = 1 << (int)NoloButtonID.Trigger;
    public const uint Menu = 1 << (int)NoloButtonID.Menu;
    public const uint System = 1 << (int)NoloButtonID.System;
    public const uint Grip = 1 << (int)NoloButtonID.Grip;
}
public class TouchMask
{
    public const uint TouchPad = 1 << (int)NoloTouchID.TouchPad;
}
```