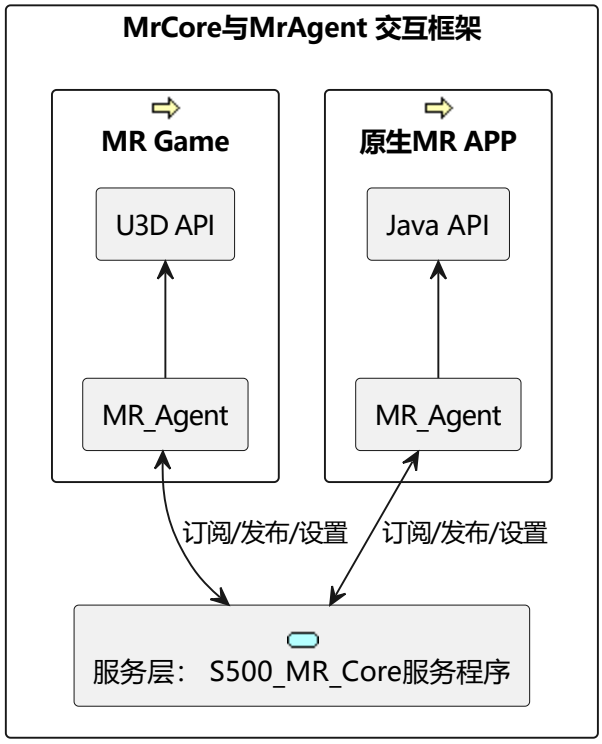
MRAgent Android 开发指南

一、 概 述

1.1 SDK基础

MRCore为SK510混合现实交互平台的核心服务组件，提供体感动作交互，语音交互等核心功能，对外可以输出用户的骨架信息、动作信息、语音指令信息、抠图信息， 为第三方游戏、MR应用程序提供方便快捷的MR交互体验。

MRAgent为游戏、MR应用程序使用MRCore的功能提供简单API接口，通过MRAgent获取用户的骨架信息、动作信息、语音指令信息、抠图信息等数据，驱动游戏和MR应用的运行。



MRAgent的API在设计上对Android和Unity3D平台保持一致，同时接口规范和使用流程与MRSDK大部分保持一致，仅在初始化部分略有不同，并且接口功能做了一些简化，便于开发者从MRSDK向MRAgent过渡。

1.2 适用范围

本SDK使用SK510产品。

二、 接入方法

2.1 开发包

开发包包含的文件说明如下：

文件夹名称	描述
/Documents/	ImiMRAgent Android SDK开发文档
/Examples/	depth, color, color&depth, 骨架， 动作示例代码
/libs/	ImiMRAgent SDK 库

库文件包含的文件说明如下：

文件名称	描述
`imimragent_java.jar`	Java封装接口实现
`imimragent_jni.so`	JNI接口模块

2.2 兼容性

系统：支持Android 5.0（API Level 21）及以上系统。
构架：支持ARM平台（提供armeabi-v7a, arm64-v8a架构的动态库）

2.3 SDK集成指南

本节将讲解如何快速的将ImiSDK集成到现有的应用中，完整的Demo请参考开发包中的示例程序。

2.3.1 拷贝库文件

Eclipse开发环境

请将开发包中的libs目录合并到工程目录的libs目录。

AndroidStudio开发环境

请将开发包中的libs/armeabi目录合并到模块的/app/jniLibs/目录下，如果没有该目录则手动新建该目，libs/ImiMR.jar合并到模块的libs目录。

2.3.2 配置AndroidManifest.xml

AndroidManifest.xml配置主要内容为：增加权限。具体示例如下：

```
<uses-permission android:name="android.permission.INTERNET" />
```

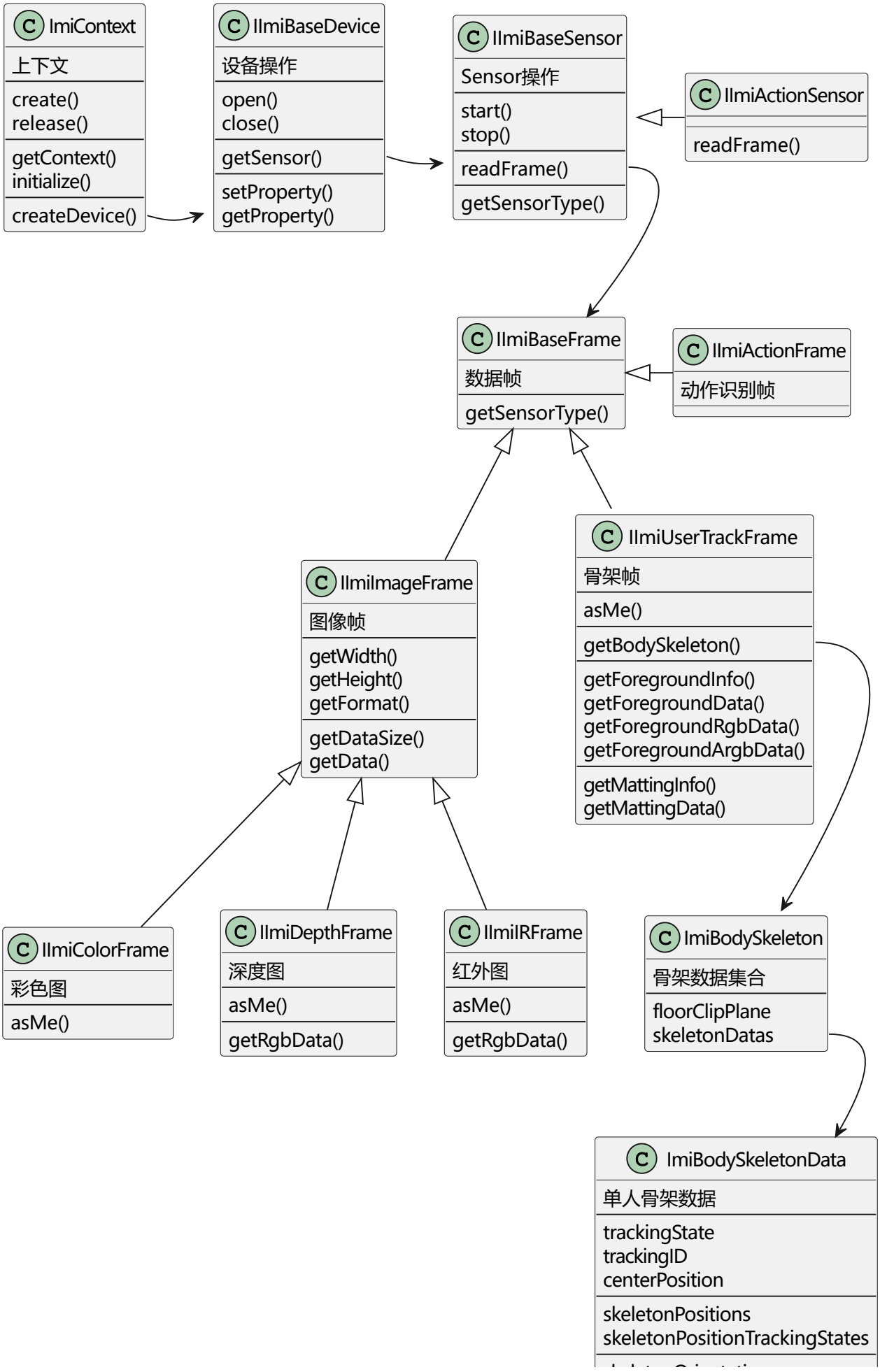
各权限含义说明如下：

名称	用途
android.permission.INTERNET	允许应用联网

三、 MRAgent的接口

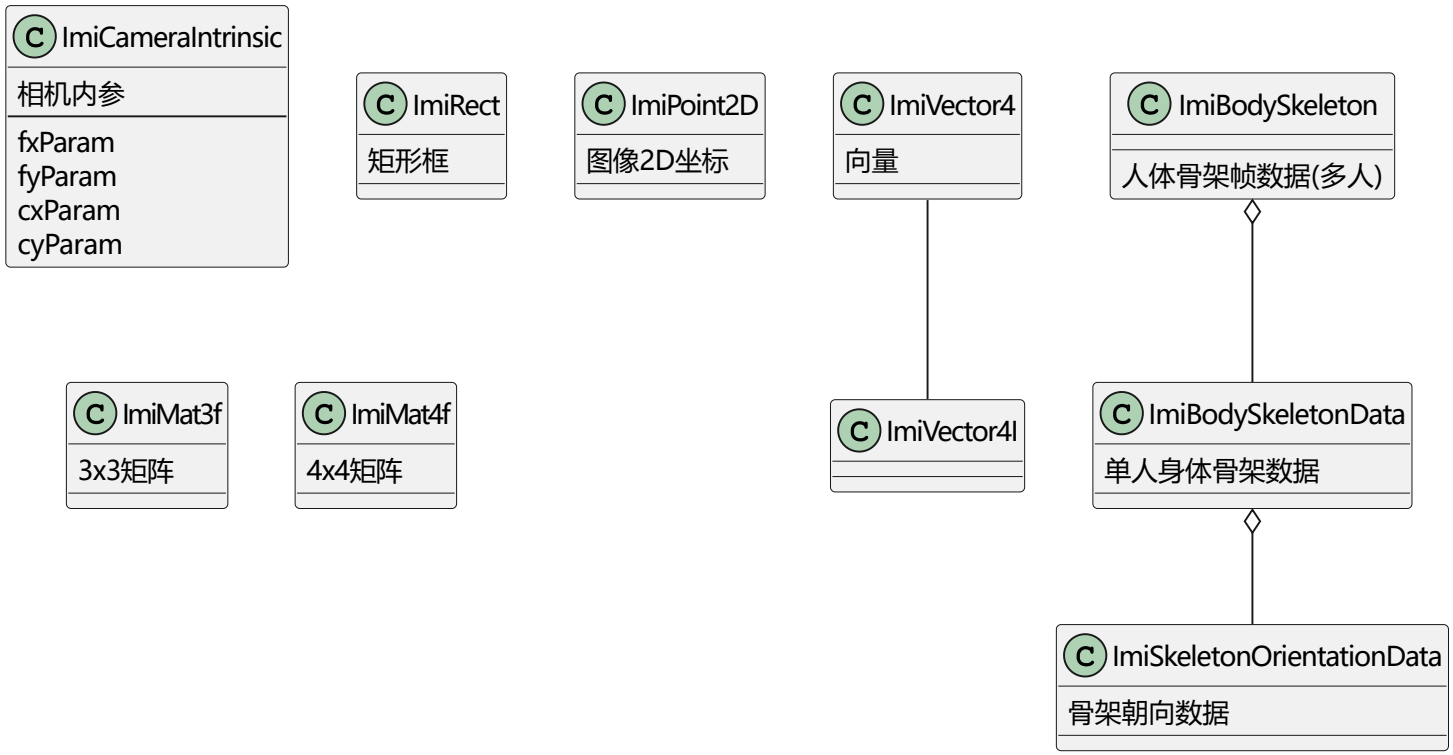
3.1 MRAgent的接口组成

主要类


ImiContext
上下文
create()
release()
getContext()
initialize()
createDevice()
IImiBaseDevice
设备操作
open()
close()
getSensor()
setProperty()
getProperty()
IImiBaseSensor
Sensor操作
start()
stop()
readFrame()
getSensorType()
IImiActionSensor
readFrame()
IImiBaseFrame
数据帧
getSensorType()
IImiActionFrame
动作识别帧
IImiImageFrame
图像帧
getWidth()
getHeight()
getFormat()
getDataSize()
getData()
IImiUserTrackFrame
骨架帧
asMe()
getBodySkeleton()
getForegroundInfo()
getForegroundData()
getForegroundRgbData()
getForegroundArgbData()
getMattingInfo()
getMattingData()
IImiColorFrame
彩色图
asMe()
IImiDepthFrame
深度图
asMe()
getRgbData()
IImiIRFrame
红外图
asMe()
getRgbData()
ImiBodySkeleton
骨架数据集合
floorClipPlane
skeletonDatas
ImiBodySkeletonData
单人骨架数据
trackingState
trackingID
centerPosition
skeletonPositions
skeletonPositionTrackingStates

skeletonOrientations
skeletonOrientationStates

数据类



3.2 ImiContext

负责MrAgent的上下文管理

3.2.1 initialize

设置与MRCore连接的参数.

```
int initialize(String filePath);
```

3.2.2 createDevice

创建IImiBaseDevice

```
IImiBaseDevice createDevice();
```

3.3 IImiBaseDevice

负责与MRCore虚拟设备的连接

3.3.1 open

打开IImiBaseSensor

```
int open(int sensorTypes);
```

3.3.2 getSensor

获取已经打开的IImiBaseSensor

```
IImiBaseSensor getSensor(int sensorType);
```

3.4 IImiBaseSensor

负责某一类型信息的获取

3.4.1 start

启动Sensor

```
int start(int startMode);
```

3.4.2 stop

停用Sensor

```
int stop();
```

3.4.3 readFrame

读取帧数据

```
IImiBaseFrame readFrame(int timeoutMs);
```

3.5 IImiBaseFrame

某类型数据帧的结构.

3.6 MRAgent的数据类型

3.6.1 彩色图数据

图像数据以ByteBuffer作为载体.

IImiColorFrame.getData

复制图像数据到 `ByteBuffer`

```
void getData(ByteBuffer rgbData);
```

IImiDepthFrame.getRgbData

将深度图数据转换为彩色图像数据并负责到 `ByteBuffer`

```
void getRgbData(ByteBuffer rgbData, int convertMode);
```

IImiIRFrame.getRgbData

将深度图数据转换为彩色图像数据并负责到 `ByteBuffer`

```
void getRgbData(ByteBuffer rgbData, int convertMode);
```

IImiUserTrackFrame.getForegroundRgbData

将深度图前景数据转换为彩色图像数据并负责到 `ByteBuffer`

```
void getForegroundRgbData(ByteBuffer rgbData, int convertMode);
```

```
void getForegroundArgbData(ByteBuffer rgbData, int convertMode);
```

3.6.2 用户身体骨架数据

用户骨架数据以 `IImiBodySkeleton` 作为载体

IImiUserTrackFrame.getBodySkeleton

```
IImiBodySkeleton getBodySkeleton();
```

3.6.3 用户动作数据

3.6.4 抠图数据

四、 实例代码

4.1 基本流程

1. APP开始时创建上下文

```
ImiContext.create(null);  //传null即可
imiContext = ImiContext.getContext();
imiContext。initialize(conf_file);
```

2. 根据上下文创建 IImiBaseDeviece

```
IImiBaseDevice mDevice = imiContext.createDevice();
mDevice.open(ImiSensorType.COLOR || ImiSensorType.USERTRACK || ImiSensorType.ACTION);
```

3. 根据 IImiBaseDeviece 创建 IImiBaseSensor

```
IImiBaseSensor colorSensor = mDevice.getSensor(ImiSensorType.COLOR);
colorSensor.start();

IImiBaseSensor userTrackSensor = mDevice.getSensor(ImiSensorType.USERTRACK);
userTrackSensor.start();

IImiActionSensor actionSensor = new IImiActionSensor(userTrackSensor);
```

4. 循环读取帧数据

```
IImiBaseFrame fr = sensor.readFrame(timeoutMs);
// todo 对Frame进行特殊控制，需要先转化为子类型
IImiColorFrame colorFr = IImiColorFrame.asMe(fr);
if(colorFr!=null){
    // todo Frame实际类型不匹配是转化不成功的。
    // 使用完Frame需要及时清理，否则缓冲区满后无法接收新的Frame
    colorFr.release();
}
```

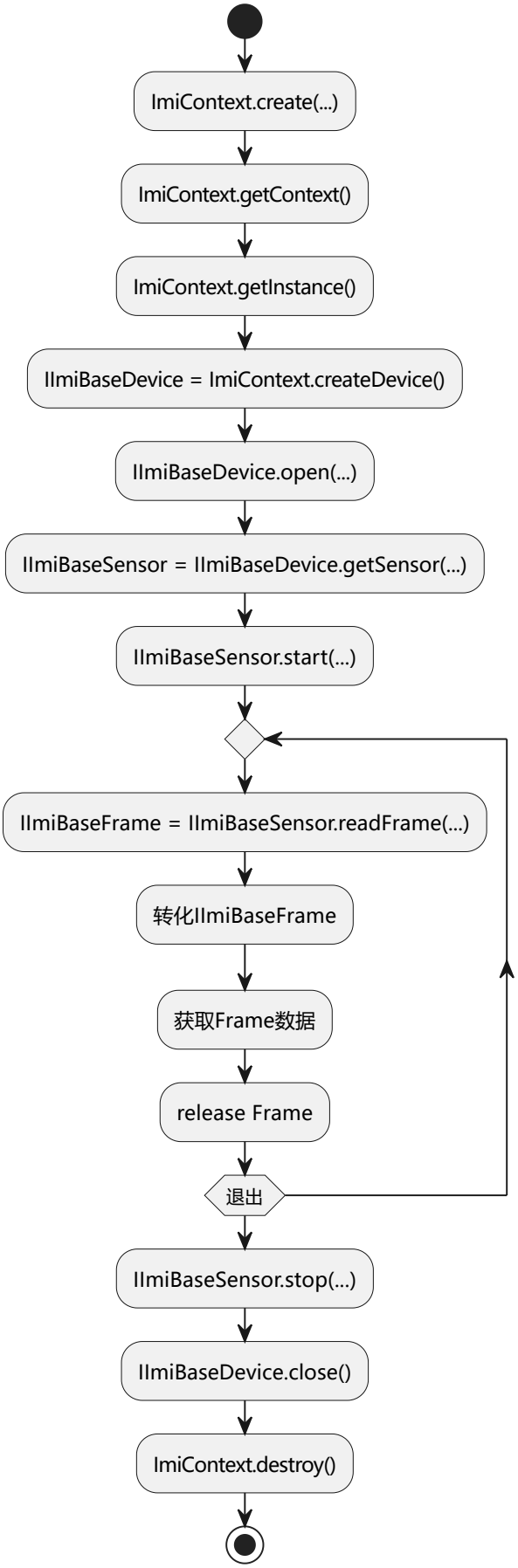
5. 关闭Sensor

```
sensor.stop();
```

6. 关闭设备，清理上下文

```
device.close();
imiContext.destroy();
```

基本代码流程图:


ImiContext.create(...)
ImiContext.getContext()
ImiContext.getInstance()
IImiBaseDevice = ImiContext.createDevice()
IImiBaseDevice.open(...)
IImiBaseSensor = IImiBaseDevice.getSensor(...)
IImiBaseSensor.start(...)
IImiBaseFrame = IImiBaseSensor.readFrame(...)
转化IImiBaseFrame
获取Frame数据
release Frame
退出
IImiBaseSensor.stop(...)
IImiBaseDevice.close()
ImiContext.destroy()

4.2 实例代码

```
//MRAgent创建
ImiContext.create(null);

ImiContext ctx = ImiContext.getContext();
// todo: 与MRSDK的区别
ctx.initialize(mragent_config_file);
// 获取列表，一般只有一个元素
ImiDeviceAttribute[] dev_list = ctx.getDeviceList().toArray();
if(dev_list.length>0){
    //
    IImiBaseDevice dev = ctx.createDevice(dev_list[0]);
    //
    if(dev!=null){
        //打开要使用的Sensor，可以用“|”组合
        dev.open(ImiSensorType.COLOR | ImiSensorType.USERTRACK | ImiSensorType.ACTION);

        IImiBaseSensor sensor1 = dev.getSensor(ImiSensorType.COLOR);
        IImiBaseSensor sensor2 = dev.getSensor(ImiSensorType.USERTRACK);
        // IImiActionSensor的创建比较特殊，和IImiUserTarckSensor绑定。
        IImiActionSensor sensor3 = new IImiActionSensor(sensor2);
        //循环读取每一帧
        while(running){
            // 获取彩色图像
            IImiBaseFrame frame1 = sensor1.readFrame(100);
            IImiColorFrame color_fr = IImiColorFrame.asMe(frame1);
            ByteBuffer color_dt = ByteBuffer.allocate(color_fr.getDataSize());
            color_fr.getData(color_dt);
            color_fr.release();

            //获取用户骨架数据
            IImiBaseFrame frame2 = sensor2.readFrame(100);
            IImiUserTrackFrame user_fr = IImiUserTrackFrame.asMe(frame2);
            ImiBodySkeleton body = user_fr.getBodySkeleton();
            ImiBodySkeletonData[] skeletons = body.getSkeletonDatas();
            // 得到骨架数据
            for(ImiBodySkeletonData one: skeletons){
                if(one.getTrackingState()==ImiSkeletonTrackState.IMI_SKELETON_TRACKED){
                    //正常骨架.显示.
                }
            }
            // 获取前景图（标记用户ID的深度图）数据。
            ImiForegroundData fg_info = user_fr.getForegroundInfo();
```

```
            ByteBuffer fg_dt = ByteBuffer.allocate(fg_info.getWidth()*fg_info.getHeight()*2);
            user_fr.getForegroundData(fg_dt);
            user_fr.release();

            // 获取动作
            IImiBaseFrame frame3 = sensor3.readFrame(100);
            IImiActionFrame act_fr = IImiActionFrame.asMe(frame3);
            // todo
            act_fr.release();
        }


        dev.close();

    }




}


ImiContext.destroy();
```

五、 常见问题及解决办法