布丁豆场景开发进阶功能

版本	日期	说明	作者
1.0	20171031	init	Lee
1.1	20171031	添加HOOOK	Lee

场景切换 （Scene Switch）

roobo OS系统提供了场景切换的通知服务，通过这个服务，可以定制自己的场景切换动画或者实现相关需求。

场景切换过程

* 获取ISceneSwitchService
* 设置场景切换回调ISceneSwitcher.Stub，场景切换有两个回调：
* startSceneSwitch()， 场景开始切换，详见注释
* stopSceneSwitch()，场景切换结束，详见注释

注：示例中并没有做Binder相关的容错处理，请自行实现。

```
private ISceneSwitchService mISceneSwitchService;

private void bindSceneSwitchService() {
    IBinder binder = RooboServiceManager.getService(mContext, "ROOBO_SC
ENE_SWITCH_SERVICE");
    if (binder != null && binder.isBinderAlive()) {
        mISceneSwitchService = ISceneSwitchService.Stub.asInterface(bin
der);
    }

    if (mISceneSwitchService != null) {
        try {
            mISceneSwitchService.setSwitcher(new ISceneSwitcher.Stub()
{
                /**
                 * 场景切换开始
                 * @param fromPkg 当前场景包名
                 * @param toPkg 要启动的场景包名
                 * @throws RemoteException
                 */
                @Override
                public void startSceneSwitch(String fromPkg, String toP
```

```
kg) throws RemoteException {

                }

                /**
                 * 场景切换结束
                 * @param toPkg 已经启动的场景包名
                 * @throws RemoteException
                 */
                @Override
                public void stopSceneSwitch(String toPkg) throws Remote
Exception {

                }
            });
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

开机自启动

Configure工程里的start_binder_config.xml里定义开机拉起的应用列表,注意scene和service的type有些不同。

```
<item name="VIDEOTALK"  package="com.roobo.videotalkservice" type="serv
ice"/>
<item name="Clock" package="com.roobo.alarm" type="scene"/>
```

多进程问题

就目前来说，一个场景对应一个APK，也对应一个进程，对于多进程的场景则需要其中一个进程A接入模块通信机制负责消息传递，另一个进程B则不接入，若进程B需要响应AI消息（AICommand），则需要自行靠IPC由进程A转发到进程B后再继续处理，若进程B需要发送消息（AIEvent），则需要自行靠IPC由进程B转发到进程A后再继续处理。

AIContext

场景可以给自己设置一个 AI 上下文，以给我们的 CloudAI 做辅助说明，虽然很多时候云端可以根据语义智能地切换上下文，但是在有用户其他交互特别是有屏幕的机器人上，用户的其他交互方式也会影响场景的切换，比如在我们即将上市的布丁豆机器人上，用户可以通过点击屏幕上的图片进入 “成语接龙” 场景，这时候云端是没有收到任何语义信息的，也不会有上下文切换，这时如果用户来了一句 “四面楚歌”，云端还以为你要聊历史呢。如果在进入 “成语接龙” 场景时在语义上传至云端时顺带一个上下文信息 “Idioms”，云端即可判断当前就在 “成语接龙” 场景，就可以正确的响应相对应的用户指令，在场景退出时，上下文信息会被清空，云端就会知道场景退出，之后的语义将与成语接龙无关。

离线命令词

1. 使用离线命令词制作工具制作离线命令，**bnf**文件为源文件，生成**bnfv6**文件，再将生成后的文件后缀改为**jpg**即可。

```
!start <open_baidu>;
<open_baidu> : !tag(PRE_TAG,百度百度!id(13016));
```

1. 将**doudou_global.bnf和doudou_global.jpg**文件复制到**ASR**项目的assets目录里
2. 修改**pre_process.xml**规则文件 以下规则的含义是：命令为13016时，输出场景为Mahjong，action为Open。13016是制作离线命令词时的id

```
<rule id="baidubaidu">
        <condition>
            <item
                opcode="equal"
                var1="$input:asr.command"
                var2="13016" />
       </condition>
       <output>
            <item>
                <target value="Mahjong" />
                <action value="Open" />
            </item>
    </output>
</rule>
```

壳应用

第三方客户如果只是想将其应用放到布丁豆上测试、观察运行情况，可通过壳应用方式加入。
1. 创建一个场景，可参考Youkuchild项目
2. 在场景里打开第三方应用
3. 退出第三方应用，可以点击屏幕上悬浮的按钮退出该场景。场景会关闭应用然后退出场景。

NOUI场景

NOUI就是场景运行过程中没有页面显示，用户看到的就是2个眼睛的Launcher。目前主控里有一个**NoUIScene**项目，场景名即是**NOUI**，现目前只有一个NOUI的场景，主控会对该场景做特殊处理。*添加新的NOUI的场景可能有问题。*与一般场景的区别主要有以下2点：

- AndroidManifest.xml中添加以下声明，在场景打开时去掉动画

```
<meta-data
        android:name="ROOBO_SCENE_FLAG"
        android:value="no_ui"/>
```

- 场景需要在场景运行时仍可进行在线ASR，配置请看下面一节介绍

场景内可在线ASR

进入场景后默认不能再进行在线ASR（可识别离线指令词），如果场景需要进行在线ASR，可将场景加入到白名单里：**SystemScene**项目的**ASRController#mOnlineRecognizeSceneList**里添加场景即可。

场景拦截ASR

进入都场景后，该场景可以直接接收用户说的话,做自定义处理。以下是**ENGLISH_CHAT**场景的示例

1. 修改**pre_process.xml**规则
   含义：当top场景是ENGLISH_CHAT，并且asr.text不为空，则触发场景ENGLISH_CHAT，action是asr_value，参数是asr.text

   注：pre_process.xml中的`asr.text`*会变成Java里的*`asr_text`

```
```

```
<rule id="english_chat">
<condition>
        <item
            opcode="equal"
            var1="$input:scene.top"
            var2="ENGLISH_CHAT" />
        <item
            opcode="not"
            var1="$input:asr.text"
            var2="$null" />
</condition>
<output>
        <item>
            <target value="ENGLISH_CHAT" />
            <action value="asr_value" />
            <params>
                <asr-text value="$input:asr.text" />
            </params>
        </item>
</output>
</rule>
```

2. 处理ASR文本,在Application的**SceneEventListener#onCommand**回调方法中处理ASR文本

```
ChatActivity.sInstance.userSend(params.getString("asr-text"));
```

HOOK

如果希望拦截后还可以继续向下传递，相当于hook，添加 `<output absorb="false">`