Cocos2dx接入

在Activity中初始化SDK，添加事件响应

使用`prj.chameleon.channelapi.cbinding.NativeChannelInterface`接口中的函数，修改Cocos2dx生成的activity，重载onResume和onPause，在onCreateView中添加初始化函数，例如

首先添加import

```
import prj.chameleon.channelapi.cbinding.NativeChannelInterface;
import android.content.Intent;
```

然后在Activity中添加事件回调的处理函数

```
    @Override
    public void onResume() {
        super.onResume();
        NativeChannelInterface.onResume(1);
    }

    @Override
    public void onPause() {
        super.onPause();
        NativeChannelInterface.onPause();
    }

    @Override
    public void onStart() {
        super.onStart();
        NativeChannelInterface.onStart(this);
    }

    @Override
    public void onStop() {
        super.onStop();
        NativeChannelInterface.onStop(this);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        NativeChannelInterface.onDestroy();
    }
```

```
    @Override
    public void onNewIntent(Intent intent) {
        super.onNewIntent(intent);
            NativeChannelInterface.onNewIntent(this, intent);
    }




    public Cocos2dxGLSurfaceView onCreateView() {
        Cocos2dxGLSurfaceView glSurfaceView = new
Cocos2dxGLSurfaceView(this);
        glSurfaceView.setEGLConfigChooser(5, 6, 5, 0, 16, 8);

        // 设置运行环境
        NativeChannelInterface.setRunningEnv(this, glSurfaceView);
        return glSurfaceView;
    }
```

修改jni/Android.mk

分别增加

`LOCAL_WHOLE_STATIC_LIBRARIES += chameleoncb`

`$(call import-add-path, $(LOCAL_PATH)/../chameleon)`

`$(call import-module,chameleoncb)`

在游戏中使用接口

之后就可以使用Chameleon提供的C++接口进行登录，支付等活动。Chameleon的C++接口分为两个部分

请求接口

接口定义在chameleon/chameleoncb/ChameleonPlatformAPI.h, 有些请求接口带有一个int参数id，这是异步请求的一个透传参数，当这个请求回应时，会带回这个参数，如果不需要传0即可。

```
namespace Chameleon {
namespace ChameleonChannelAPI{
    /**
     * 设置java的虚拟机实例
     */
    void setJavaVM(JavaVM * vm);

    /**
     * 释放设置的java的虚拟机实例
     */
```

```
    void releaseJavaVM(JavaVM * vm);

    /**
     * 注册事件的回调实例
     */
    void registCallback(ChannelAPICallbackInf * callbackImp);

    /**
     * 发起游客登录，有些平台不支持游客登录或者平台一些登录策略，也有可能
     * 会发起正式的登录
     * @param {int} id, 标识该请求的id
     * @return {int}, 0表示异步请求成功，否则表示失败
     */
    int loginGuest(int id);

    /**
     * 如果是游客登录，那么通过这个函数可以发起正式注册的请求
     * @param {int} id, 标识该请求的id
     * @param {std::string} tips, 有些平台可以在注册页面显示一段提供的tips
     * @return {int}, 0表示异步请求成功，否则表示失败
     */
    int registGuest(int id, const std::string & tips);

    /**
     * 发起平台登录请求
     * @param {int} id, 标识该请求的id
     * @return {int}, 0表示异步请求成功，否则表示失败
     */
    int login(int id);

    /**
     * 如果游戏使用了二级货币，那么可以通过这个函数发起充值的请求
     * @param {int} id, 标识该请求的id
     * @param {std::string} orderId, 此次购买的ID
     * @param {std::string} uidInGame, 玩家在游戏中的id
     * @param {std::string} userNameInGame, 玩家在游戏中的名字
     * @param {std::string} serverId, 玩家所在server的ID
     * @param {std::string} currencyName, 二级货币的名称
     * @param {std::string} payInfo, 从chameleon server获得了额外支付信息
     * @param {int} rate, 二级货币的兑换比率，例如，如果1RMB可以兑换10二级货币，那么
rate=10
     * @param {int} realPayMoney, 希望玩家支付的数量，如果允许玩家自己输入，那么这个
param可能
     *                              会被忽略
     * @param {bool} allowUserChange, 是否允许玩家自己输入
     * @return {int}, 0表示异步请求成功，否则表示失败
     */
    int charge(int id,
               const std::string & orderId,
               const std::string & uidInGame,
               const std::string & userNameInGame,
```

```
                const std::string & serverId,
                const std::string & currencyName,
                const std::string & payInfo,
                int rate,
                int realPayMoney,
                bool allowUserChange);

    /**
     * 如果游戏使用了购买道具的功能，那么可以通过这个函数发起购买的请求
     * @param {int} id, 标识该请求的id
     * @param {std::string} orderId, 此次购买的ID
     * @param {std::string} uidInGame, 玩家在游戏中的id
     * @param {std::string} userNameInGame, 玩家在游戏中的名字
     * @param {std::string} serverId, 玩家所在server的ID
     * @param {std::string} productName, 道具的名称
     * @param {std::string} productId, 产品的ID
     * @param {std::string} payInfo, 从chameleon server获得了额外支付信息
     * @param {int} productCount, 购买的数量
     * @param {int} realPayMoney, 希望玩家支付的数量，如果允许玩家自己输入，那么这个
param可能
     *                              会被忽略
     * @return {int}, 0表示异步请求成功，否则表示失败
     */
    int buy(int id,
            const std::string & orderId,
            const std::string & uidInGame,
            const std::string & userNameInGame,
            const std::string & serverId,
            const std::string & productName,
            const std::string & productId,
            const std::string & payInfo,
            int productCount,
            int realPayMoney);

    /**
     * 该平台是否支持账号切换
     * @return {bool}, 是否支持账号切换
     */
    bool isSupportSwtichAccount();

    /**
     * 发起切换账户的请求
     * @param {int} id, 标识该请求的id
     * @return {int}, 0表示异步请求成功，否则表示失败
     */
    int switchAccount(int id);

    /**
     * 创建并且显示平台的工具条
     * @param {int} position, 位置参数
     * @return {int}, 0表示异步请求成功，否则表示失败
```

```
 */
int createAndShowToolbar(int position);

/**
 * 显示或者隐藏工具条
 * @param {bool} isVisible, 显示或者隐藏
 */
void showToolbar(bool isVisible);

/**
 * 销毁工具条
 */
void destroyToolbar();

/**
 * 查询防沉迷信息
 * @param {int} id, 标识该请求的id
 * @return {int}, 0表示异步请求成功, 否则表示失败
 */
int antiAddiction(int id);

/**
 * 退出渠道SDK
 * @return {int}, 0表示异步请求成功, 否则表示失败
 */
int exit();

/**
 * 获取渠道的名字
 * @return {string}
 */
std::string getChannelName();

/**
 * 获取玩家的id
 * @return {string}
 */
std::string getUid();

/**
 * 获取玩家的token
 * @return {string}
 */
std::string getToken();

/**
 * 获取玩家的pay token
 * @return {string}
 */
std::string getPayToken();
```

```
    /**
     * 响应从Chamemleon SDK服务器回来的信息
     * @param {string} loginRsp SDK服务器的回包
     * @return {bool} 登陆是否验证登陆成功
     */
    bool onLoginRsp(const std::string & loginRsp);

     /**
     * 提交用户信息
     * @param roleId 用户在游戏中的ID
     * @param roleName 用户在游戏中的名
     * @param roleLevel 用户等级
     * @param zoneId zone id
     * @param zoneName zone的名称
     */
    bool submitPlayerInfo(const std::string & roleId,
                          const std::string & roleName,
                          const std::string & roleLevel,
                          int zoneId,
                          const std::string & zoneName);


    /**
     * 是否登录
     */
    bool isLogined();
}
}
```

回调接口

您必须实现一个这样的接口，然后由请求接口中的`registCallback`注册进chameleon的SDK。这个接口主要用来响应SDK的请求的回调，或者渠道SDK自动触发的一些信息。

chameleoncb/CPlatformAPICallback.h

```
class ChannelAPICallbackInf {
public:
    virtual ~ChannelAPICallbackInf() {}
    /**
     * 玩家将要进行切换账户，这个函数只会在玩家使用平台的activity时候由平台触发的
     * 回调，一般在这个时候做一些玩家数据保存工作，当前玩家马上就要登出了
     */
    virtual void preAccountSwitch() {}
    /**
     * 玩家已经完成了账户切换
     * @param {int} code，表示玩家是否已经登录了新的账户，如果是，那么loginInfo则为有效的数据
     * @param {string} loginInfo，如果登录了新的账户，那么这个是新账户的登录信息
```

```
     */
    virtual void afterAccountSwitch(int code, const std::string * loginInfo)
{}
    /**
     * 当前用户从平台退出的回调
     */
    virtual void onAccountLogout() {}
    /**
     * 如果当前玩家是游客登录，并且通过平台的activity注册或者登录了，那么
     * 会通过这个回调告诉游戏正式的玩家登录信息
     * @param {string} loginInfo, 玩家的登录信息
     */
    virtual void onGuestBind(const std::string & loginInfo) {}

    /**
     * 由Chameleon SDK主动发起的登录流程，如果玩家是由游客身份登录，那么就会回调这个函数
     * @param {int} id, 对应请求的ID
     * @param {int} code, 登录成功或者失败
     */
    virtual void onLoginGuest(int id, int code) {}

    /**
     * 由Chameleon SDK主动发起的注册游客的请求的回调
     * @param {int} id, 对应请求的ID
     * @param {int} code, 注册或者登录成功与否
     * @param {string} loginInfo, 玩家的登录信息
     */
    virtual void onRegistGuest(int id, int code, const std::string *
loginInfo) {}

    /**
     * 由Chameleon SDK主动发起的登录流程，如果玩家由平台正式用户登录，则会回调这个函数
     * @param {int} id, 对应请求的ID
     * @param {int} code, 登录成功或者失败
     * @param {string} loginInfo, 玩家的登录信息
     */
    virtual void onLogin(int id, int code, const std::string * loginInfo) {}

    /**
     * 充值二级货币的请求的结果回调
     * @param {int} id, 对应请求的ID
     * @param {int} code, 登录成功或者失败
     */
    virtual void onCharge(int id, int code) {}

    /**
     * 购买道具的请求的结果回调
     * @param {int} id, 对应请求的ID
     * @param {int} code, 登录成功或者失败
     */
    virtual void onBuy(int id, int code) {}
```

```
    /**
     * 由Chameleon SDK主动发起的切换账户的请求的结果回调，无论结果如何，最好当做当前的
用户
     * 已经登出了
     * @param {int} id，对应请求的ID
     * @param {int} code，登录成功或者失败
     * @param {string} loginInfo，玩家的登录信息
     */
    virtual void onSwitchAccount(int id, int code, const std::string *
loginInfo) {}

    /**
     * 玩家通过点击平台的toolbar可能会切换到平台的activity中去，这个回调用于
     * 通知玩家进入或者退出了平台的activity
     * @param {int} flag，参见ChameleonErrorCode.h中
     *                    CHAMELEON_SDK_UI_*
     */
    virtual void onToolbar(int flag) {}

    /**
     * 如果调用了PlatformAPI中的onPause函数，那么会进入平台的pause事件处理函数，
     * 当平台的pause事件处理完之后，会回调这个函数
     */
    virtual void onResume() {}

    /**
     * 防沉迷信息请求的结果回调
     * @param {int} id，对应请求的ID
     * @param {int} code，登录成功或者失败
     * @param {int} flag，参见ChameleonErrorCode.h中
     *                    CHAMELEON_ANTIADDICTION_*
     */
    virtual void onAntiAddiction(int id, int code, int flag) {}

    /**
     * 调用销毁SDK实例的结果回调
     * @param {int} id，对应请求的ID
     */
    virtual void onExit() {}

    virtual void onRunProtocol(int id, int code, const std::string &
protocol,
      const std::string & message) {}
};
```

编译

先使用Cocos2dx的脚本编译好so文件，之后使用客户端工具即可打包各个渠道