Android编译步骤

库编译

1. 安装go 环境，设置PATH和GO_ROOT
2. 下载Android NDK并解压缩，压缩，https://dl.google.com/android/repository/android-ndk-r19c-windows-x86_64.zip
3. 安装mingw-w64-install.exe 完成后进入安装目录，如 C:\Program Files\mingw-w64\x86_64-7.3.0-posix-seh-rt_v5-rev0\mingw64\bin
   复制mingw32-make.exe 一份改名为 make.exe，同时将mingw的安装bin目录 加入PATH 如 C:\Program Files\mingw-w64\x86_64-7.3.0-posix-seh-rt_v5-rev0\mingw64\bin 加入PATH环境变量
4. 安装MSYS-1.0.11.exe, 安装时会选择mingw32目录，可以在C盘建立一个目录如C:\mingw 软连接到 mingw的安装目录。
   将msys的安装bin目录 加入PATH 如 C:\msys\1.0\bin
5. 运行 android-studio 配置里面sdkmanager 安装android sdk （Android 9.0）和 ndk （安装ndk在sdk manager 的 SDK tools），ndk安装19.2.5345600
   配置环境变量ANDROID_HOME=C:\Users\xb\AppData\Local\Android\Sdk 和 ANDROID_NDK=C:\Users\xb\AppData\Local\Android\Sdk\ndk-bundle
6. 安装jdk jdk-8u201-windows-x64.exe ，配置环境变量 JAVA_HOME=C:\Program Files\Java\jdk1.8.0_201
   并且将javasdk的bin目录加入PATH环境变量C:\Program Files\Java\jdk1.8.0_201\bin
7. 使用 git@coding.xbonline.net:blockchain/go-ethereum.git上的分支代码，当前为develop分支
8. 执行 make android_gengine 会失败在 build/env.sh，执行一次后
   修改 env.sh中 18行 ln -s ../../../../../. go-ethereum 注释掉这行 再执行 make android_gengine
9. make android_gengine 还是会失败，修改build/ci.go中doAndroidGengineArchive 函数中 796 797 798 三行 注释掉
   //if os.Getenv(“ANDROID_NDK”) == “” {
   // log.Fatal(“Please ensure ANDROID_NDK points to your Android NDK”)
   //}
   同时将
   802行 build.MustRun(gomobileTool(“init”, “–ndk”, os.Getenv(“ANDROID_NDK”)))
   修改为
   build.MustRun(gomobileTool(“init”)) 即可
10. 执行make android_gengine 会在build/bin下生成gengine.aar

项目编译

安装cocos creator 2.2.2，会提示安装vs2017，安装vs2017

1. 去git上正确代码分支，目前是RMT

2. 用cocos creator 2.2.2 打开目录，构建项目，选择 android，andoird28， armv7 和armv8 会生成 build目录
   (下面修改步骤可以参考脚本android_modify.bat)
3. 将build_kaleido/jsb-link/frameworks/runtime-src/proj.android-studio/app/src/org/cocos2dx/javascript/AppActivity.java中的修改拷贝至build/jsb-link/frameworks/runtime-src/proj.android-studio/app/src/org/cocos2dx/javascript/AppActivity.java
4. 将build_kaleido/jsb-link/frameworks/runtime-src/proj.android-studio/app/src/io整个目录拷贝至build/jsb-link/frameworks/runtime-src/proj.android-studio/app/src/
5. 将build_kaleido/jsb-link/frameworks/runtime-src/proj.android-studio/app/res/整个目录拷贝至build/jsb-link/frameworks/runtime-src/proj.android-studio/app/res/
6. 比较build_kaleido/jsb-link/frameworks/runtime-src/proj.android-studio/app/proguard-rules.pro和build/jsb-link/frameworks/runtime-src/proj.android-studio/app/proguard-rules.pro，修改使其相同，修改点在文件最后添加:

```
 # keep kaleido for release.
-keep public class io.kaleidochain.** { *; }
-dontwarn io.kaleidochain.**
```

7. 比较build_kaleido/jsb-link/frameworks/runtime-src/proj.android-studio/app/build.gradle和build/jsb-link/frameworks/runtime-src/proj.android-studio/app/build.gradle，
   修改点为将 minifyEnabled 和 shrinkResources 均设置为 false，和

```
        applicationId "io.kaleidochain.BlockPoker"
        minSdkVersion PROP_MIN_SDK_VERSION
        targetSdkVersion PROP_TARGET_SDK_VERSION
        versionCode 2
        versionName "1.0.1"
```

和

```
        buildTypes {
        release {
            debuggable false
            jniDebuggable false
            renderscriptDebuggable false
            minifyEnabled false
            shrinkResources false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
            if (project.hasProperty("RELEASE_STORE_FILE")) {
                signingConfig signingConfigs.release
            }
```

8. 对比 \build_kaleido\jsb-link\frameworks\runtime-src\proj.android-studio\app\AndroidManifest.xml 和

\build_kaleido\jsb-link\frameworks\runtime-src\proj.android-studio\app\AndroidManifest.xml 修改点：

```
package="io.kaleidochain.BlockPoker"
...
android:keepScreenOn="true"
```

9. 在build/jsb-link/frameworks/runtime-src/proj.android-studio/app/中创建libs目录，并将gengine.aar 拷贝到libs目录。
10. 重新构建项目，选择 android ， andoird28， armv7 和armv8， 编译会生成最后apk包，apk包在 build\jsb-link\frameworks\runtime-src\proj.android-studio\app\build\outputs\apk 目录

    注：如果在windows编译中修改了cocos 的安装目录的 如 D:\CocosCreator_2.2.2\resources\cocos2d-x\cocos\platform\CCApplication.h， 修改重点是：

```
inline std::shared_ptr<Scheduler> getScheduler() const { return _scheduler; }
改为
inline Scheduler* getScheduler() const { return Application::_scheduler.get();}

    static std::shared_ptr<Scheduler> _scheduler;
改为
public :
static std::shared_ptr<Scheduler> _scheduler;
```

这样修改后在android编译也可能出现_scheduler = Application::getInstance()->getScheduler()的编译错误;，需要对对应的代码做修改： `_scheduler = Application::getInstance()->getScheduler();` 这个地方的全部改成 `_scheduler = Application::_scheduler;`
主要是 cocos目录下 D:\CocosCreator_2.2.2\resources\cocos2d-x\cocos\network文件 HttpClient-android.cpp