LGame-Android-0.3.0版中文使用文档

项目地址：http://code.google.com/p/loon-simple

开源协议：Apache License 2.0

文档历史记录

2010-09-15 LGame-0.2.8 使用文档（第一版）发布。
2010-12-11 LGame-0.2.9 使用文档发布。
2011-05-25 LGame-0.3.0 使用文档发布。

LGame-Android 版为 LGame-J2SE（PC 版）的手机（Android）精简版本，除辅助功能较 LGame-J2SE 版为少外，主要类及函数与 LGame-J2SE 版完全相同。

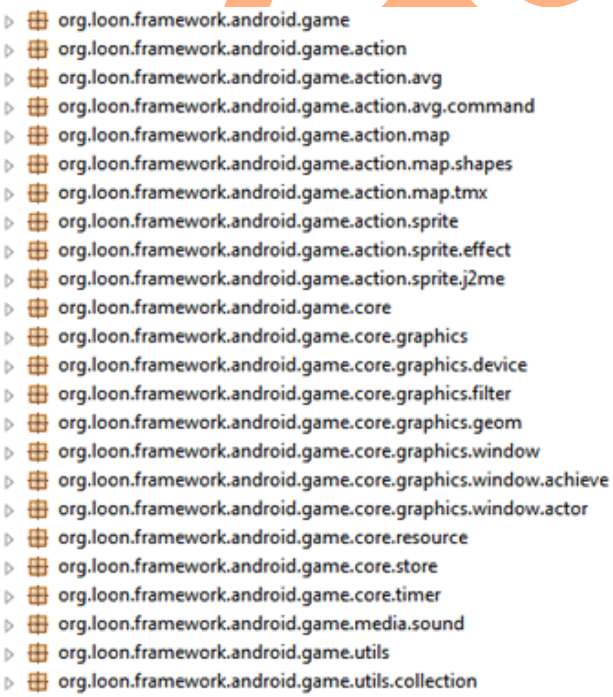
相较其它引擎而言，LGame 具有很高的 Android 平台兼容性，可以运行在所有支持 Android1.5（最低要求）及以上版本的手机环境当中。此外，绝大多数 Android 平台衍生系统（例如某些国内“自主研发”的）也可以支持 LGame-Android 正常运转。

截止到 LGame-Android-0.3.0 版，LGame-Android 游戏引擎共包含代码近九万行，主要功能从底层图形接口到游戏控制、物理引擎、重力感应、资源存储、资源加密、地图绘制、精灵绘制、组件绘制、脚本操作、AVG 模式支持、触摸屏模拟按键支持（详见 EmulatorButton 部分）、音频操作、以及字符、数字、密码等辅助操作功能也尽皆涵盖。

而预计提供的 AVG、SRPG(SLG)、RPG、STG、ACT、PUZ、FTG、RTS 等八种“专属游戏类型开发扩展包”，也已经提供有 AVG 及 SRPG 两种，充分保证用户能够在近乎“零编码”的情况下完成特定扩展包所对应的游戏类型开发（预计在 0.4 发布前会追加 RPG、STG、ACT、PUZ 等四种扩展包）。

应该说，LGame 在设计上极大简化了原本繁琐的 Android 游戏开发流程，只要您正使用 LGame-Android 引擎进行游戏构建，便能够在不使用任何第三方组件的前提下，完成任意您想要实现的 Android2D 游戏（而且，还可以非常轻松地将它移植到 PC 环境，发布个试玩版之类。另外，支持 WP7 环境的 C#语法移植已经启动，预计 2011 年 11 月前完成）。

LGame-Android-0.3.0 版 Core 包结构如下所示：



此文档将顺序逐一讲解 LGame 相关类及函数的作用与功效。PS：以下说明中子类暂不包含父类函数以及构造方式，具体请参考涉及的父类函数以及相关源码中的构造方法。

1、org.loon.framework.android.game

此包下共有类 7 个，分别为 Android2DHandler，Android2DSurface，IAndroid2DHandler，IHandler，Location，LGameAndroid2DActivity，LGameAndroid2DView，其中最关键的两个类在于 LGameAndroid2DActivity 以及 LGameAndroid2DView，它们将承载起 LGame 引擎启动及画面展示的主要功效。

（1） LGameAndroid2DActivity

LGameAndroid2DActivity 继承自 Activity，作为子类拥有 Activity 的一切功能，它的作用在于初始化 LGame 以及提供基本的引擎初始参数设置和将标准 Android 视图、按键转换到 LGame 框架当中。

该类提供有函数如下所示：

函数名	函数说明
onMain()	这是一个抽象的初始化函数，使用者必须在 LGameAndroid2Dactivity 子类中实现 onMain 才能完成 LGame 引擎的初始化。该函数多用来进行横竖屏设定，调节界面尺寸，设定 Admob 广告信息，创建游戏窗体实例等综合操作。
maxScreen(int w,int h)	此函数用来设定游戏窗体大小，当手机画面小于或等于此设定时，此设定将被完全执行，否则将以画面允许的最大比值执行。它的出现位置必须保持在所有其它设定出现之前，否则此设定将不被执行。在不设置此项时，LGame 默认的游戏窗体大小为 480x320（横），320x480（纵）。
Initialization(……)	Initialization 是 LGame 的游戏窗体及 Admob 广告初始用函数，当 Initialization 为 True 时，游戏将以横屏显示，否则以纵屏显示，它的出现位置应该保持在 maxScreen 设定之后，其它设定出现之前。此外，在 Initialization 中也可以顺序设置 Admob 广告是否显示，广告的出现位置，广告的标识 ID，广告的关键字，以及广告的刷新频率。
setScreen(IScreen screen)	设定一个 LGame 的游戏窗体，以供游戏运行。只有继承自 Screen 或者实现 IScreen 的相关类，才能被此函数执行。
setFPS(long frames)	设定游戏画面刷新速度，该设定只对默认的标准 Screen 类有效。
setShowFPS(boolean flag)	设定是否显示画面刷新速度，该设定只对默认的标准 Screen 类有效。
setLogo(LImage img)	设定游戏初始 logo 图片，此图片只有在设定显示 Logo 时才会被执行。
setShowLogo(boolean	设定游戏初始 logo 是否被显示。

showLogo)	
setDebug(boolean debug)	设定 debug 模式，此模式仅供无调试环境的真机使用，当此参数为真时，程序异常时会弹出异常提示。
openAssetResource(String fileName)	获得指定名称的 assets 文件夹下资源。
getOrientation()	获得当前屏幕方向。
isLandscape()	获得游戏是否进入横屏。
isKeyboardOpen()	获得游戏中是否显示了小键盘。
showAndroidProgress(String title,String message)	弹出一个指定内容的进度条。
showAndroidAlert(String title, String message)	弹出一个指定内容的提示框。
showAndroidSelect(String title, String text[])	弹出一个指定内容的选择框。
showAndroidException(Exception ex)	弹出一个指定内容的异常信息。
getAndroidSelect()	返回打开的选择框选中项。
openBrowser(String url)	打开浏览器并访问指定页面。
showAD()	显示 Admob 广告。
hideAD()	隐藏 Admob 广告。
getActivity()	返回当前游戏的 Activity 实例。
getVersionCode()	获得当前游戏版本号。
getVersionName()	获得当前游戏版本名。
getPackageInfo()	获得当前游戏环境信息集合。
getScreenDimension()	获得当前手机窗体实际大小。
getFrameLayout()	获得当前游戏的 FrameLayout 布局，您可以在此基础上任意添加其它 Android 原生组件，而不会对游戏产生任何干扰。
showScreen()	显示游戏画面，在运行此项前游戏将无法显示。

（2） LGameAndroid2DView

LGameAndroid2DView 是 SurfaceView 的子类，它的作用在于和 LGameAndroid2Dactivity 相互配合，最终完成 LGame 引擎游戏窗体的绘制与显示。

该类提供有函数如下所示：

函数名	函数说明
setScreen(IScreen screen)	设定一个 LGame 的游戏窗体，以供游戏运行。只有继承自 Screen 或者实现 IScreen 的相关类，才能被此函数执行。
setEmulatorListener(EmulatorListener emulator)	设定 LGame 模拟按键所使用的监听器，当此项不为 NULL 时，模拟按键将被显示于游戏画面之上。另外，如果 Screen 实现了 EmulatorListener，此函数会被自动调用。
getEmulatorListener()	返回当前的模拟按键监听器。

getEmulatorButtons()	返回当前的模拟按键集合。
getAndroid2DBitmap()	返回当前的游戏画面(Bitmap)。
getAndroid2DImage()	返回当前的游戏画面(LImage)
destroyView()	注销游戏窗体，并强制令 LGame 视图系统停止运作。
update()	强制刷新游戏画布。
update(……)	强制刷新图形到游戏默认位置(0,0)，图形可以是 LImage 或 Bitmap 格式。
update(……,int w,int h)	强制刷新指定图形到游戏中，并按照提供的 w,h 参数按比例居中显示，图形可以是 LImage 或 Bitmap 格式。
updateLocation(……,int x, int y)	强制刷新指定图形到游戏指定位置，图形可以是 LImage 或 Bitmap 格式。
updateFull(……,int w,int h)	强制刷新指定图形到游戏中，并将其扩大为 w,h 参数所指定的大小，然后按比例居中显示。图形可以是 LImage 或 Bitmap 格式。
updateResize(……,int w,int h)	强制刷新指定图形到游戏中，并按照 w,h 扩大或缩小为指定大小，然后按比例居中显示。图形可以是 LImage 或 Bitmap 格式。
setShowLogo(boolean showLogo)	设定是否显示 logo
setLogo(LImage img)	设定游戏 logo 图形
setDebug(boolean debug)	设定是否开启 debug 模式
setFPS (long frames)	设定游戏的 FPS（仅对标准 Screen 有效）
setShowFPS(boolean isFPS)	设定是否显示游戏 FPS（仅对标准 Screen 有效）
setRunning(boolean running)	设定是否允许 LGameAndroid2DView 中游戏主线程继续运行。
setDrawPriority(int drawPriority)	设定 LGameAndroid2DView 中游戏主线程运行模式。
getGameHandler()	返回当前游戏句柄。（内含 LGameAndroid2Dactivity，LGameAndroid2DView，还有一些其它重要配置结果）

2、org.loon.framework.android.game.action.avg

此包下共有类 5 个，分别为 AVGScreen.、AVGScript、CG、Chara、MessageDialog，其中最要的是 LGame 四大 Screen 之一的 AVGScreen。

（1） AVGScreen

AVGScreen 是 LGame 引擎的四种 Screen 之一，与其它 Screen 的主要差异在于，它直接继承自 CanvasScreen（CanvasScreen 继承自标准 Screen），且必须要注入脚本才能运作，该类默认为自动刷新画面（但允许调用父类提供的 repaint 函数）。

函数名	函数说明
nextScript()	强制命令脚本向下解析一行，如果无条件调用此参数，则会出现自动播放效果。
nextScript(String message)	此为抽象函数，需要继承 AVGScreen 的子类进行实现，每次 AVGScreen 进行脚本解析时，该行脚本数据会自动向此

	函数发送此行脚本内容并触发此函数运行。
onSelect(String message, int type)	此为抽象函数，需要继承 AVGScreen 的子类进行实现，当 AVGScreen 中出现选择框，并且用户进行选择时。有关于选择框的信息及选中项会自动传送到此函数中。
initMessageConfig(LMessage message)	此为抽象函数，并且仅在 AVGScreen 初始化时有效，此函数用以修正游戏消息框的设置为指定参数，否则将以默认参数运行 AVG 模式。
initSelectConfig(LSelect select)	此为抽象函数，并且仅在 AVGScreen 初始化时有效，此函数用以修正游戏选择框的设置为指定参数，否则将以默认参数运行 AVG 模式。
initCommandConfig(Command command)	此为抽象函数，并且仅在 AVGScreen 初始化时有效，此函数用以修正游戏脚本的设置为指定参数，否则将以默认参数运行 AVG 模式。
initCommandConfig(String fileName)	初始化指定的游戏脚本到 AVGScreen 中，如果在 AVGScreen 中想要更换脚本，可以使用此函数调用。
drawScreen(LGraphics g)	此为抽象函数，当实现此函数后，可以对 AVG 模式下游戏画面进行底层绘制。
select(int type)	强制 AVG 脚本选择指定的选择项。
getSelect()	获得当前 AVG 脚本所在选择项内容。
add(……)	为 AVGScreen 添加一个组件或者精灵。
click ()	触发 AVG 模式下游戏点击事件。
setLocked(boolean locked)	锁定脚本事件触发，当此方法参数为 True 时，默认的 click 方法无效。可以通过锁定点击事件，来制作特殊的 AVGScreen 窗体（比如商店界面，自定义一些商品组件插入，而后锁定脚本，当购买指定物体后才触发新的脚本）
isLocked()	判定当前 AVGScreen 脚本事件触发是否遭到了锁定。
isScrFlag()	判断当前命令行是否被打上了标准脚本标记。（有此标记行脚本的下一行需要点击屏幕才能进入）
getSelectMessage()	获得当前选择框的标题信息。
isCommandGo()	获得当前脚本是否正在运行。
getMessage()	获得当前 AVG 模式下 LMessage 组件的实例。
getLSelect()	获得当前 AVG 模式下 LSelect 组件的实例。
getCommand()	获得当前 AVG 模式下 Command 脚本控制器的实例。
getScriptName()	获得当前 AVG 模式下脚本名称。
getScrCG()	获得当前 AVG 模式下角色 CG 控制器。
getDesktop()	获得当前 AVG 模式下组件控制器。
getSprites()	获得当前 AVG 模式下精灵控制器。
pointerPressed(double x, double y)	点击触摸屏。
pointerMove(double x, double y)	移动触摸屏。
pointerReleased(double x, double y)	释放触摸屏。

keyPressed(int keyCode)	键盘按下。
keyReleased(int keyCode)	键盘放开。
setRunning(boolean running)	设定当前游戏线程是否继续运行。
setDelay(int delay)	设定当前游戏线程刷新速度。
getDialogImage()	返回当前游戏消息框图像。
getSleep()	返回当前游戏画面暂停时间。
onExit()	当脚本中调用“exit”命令时，此函数会被同时触发，多用于在此时执行 setScreen 等命令切换窗体或结束游戏。

以下为 AVGScreen 类默认支持的脚本命令，将以下命令写入任意文本文件并导入 AVGScreen，即可被 AVGScreen 解释并执行。

PS：LGame 中脚本命令不区分大小写。

脚本命令	命令说明
Include	作用： 在当前脚本中载入另外一个脚本的内容，当导入脚本结束时将继续读取当前脚本。 写法： Include res/myscript.txt

If……else……endif	作用：
	分支判断脚本流程。
	写法：
	if a>b
	XXXX
	else if a==b
	XXXX
	else
	XXXX
	endif

begin……end	作用： 此命令用以构造一组脚本，但不立即执行，仅在调用 call 命令时才能触发此命令内部的脚本。 写法： Begin showMessage Mescolor red Mes 孔曰成仁，孟曰取义。 XXXXX End
Call	作用： 此命令用以调用指定的命令段，需要和 begin……end 命令配合使用。 写法： Call showMessage

	此时 showMessage 命令段将被执行。
In……out	作用： 这是一组从输入到输出的命令，在两组命令之间的字符串数据将被以选择框形式展现给用户。 写法： In A.景德镇瓷器 B.景德镇戏剧 C.景德镇餐具 Out
select	作用： 为选择框加上标题，如果 select 命令为空，则 in……out 命令输出的选择框将只有选项，而没有标题。 写法： Select 景德镇盛产什么？ In A.景德镇瓷器 B.景德镇戏剧 C.景德镇餐具 Out
selleft	作用： 设定选择框文字在显示时距离选择框左侧的偏移距离，用以调节文字位置。 写法： seleft 10
seltop	作用： 设定选择框文字在显示时距离选择框顶端的偏移距离，用以调节文字位置。 写法： Seltop 10
mes	作用： 显示一组对话，用以令游戏角色向用户传递信息。 写法： 1、单纯显示 Mes 吃了吗？没吃？！……没吃回家吃去吧。 2、强制换行 Mes 吃了吗？\n 没吃？！……\n 没吃回家吃去吧。 3、转换指定范围内文字颜色 Mes <r 吃了吗/>？\n 没吃？！……\n 没吃回家吃去吧。 (r 为红，w 为白，b 为黑，o 为橘黄)
meslen	作用： 规定 mes 命令在显示对话时每行最多的显示字数，超过将自动换行（不填时以系统判断的默认行数执行）。 写法：

	Meslen 20
mescolor	作用： 规定 mes 命令在显示对话时文字的默认颜色。 写法： Mescolor yellow
mestop	作用： 规定 mes 命令在显示时距离消息框上方的距离，用以调节文字位置。 写法： Mestop 10
mesleft	作用： 规定 mes 命令在显示时距离消息框左边框的距离，用以调节文字位置。 写法： Mesleft 10
messtop	作用： 关闭当前画面上的消息框或选择框。 写法： messtop
gb	作用： 显示指定的背景画面。 写法： gb res/background.png
cg	作用： 显示指定的角色画面。 写法： 1、 单纯显示 cg res/role.png 2、 显示在指定位置 cg res/role.png 50 50 3、 替换当前 cg cg res/role.png to res/role1.png 4、 删除指定 cg cg del res/role.png (ps:只填写 cg del 则删除全部 cg)
sleep	作用： 令画面延迟指定时间，此段时间内画面将无响应。 写法： Sleep 1000
flash	作用： 令画面以指定颜色闪烁。 写法： Flash 200,125,200
cgwait	作用： 暂停当前画面，当点击时继续运行。

	写法： Cgwait
Fadein	作用： 以指定颜色进行屏幕淡入。 写法： Fadein black
Fadeout	作用： 以指定颜色进行屏幕淡出。 写法： Fadeout black
shake	作用; 让画面产生模拟震荡。 写法： Shake 30
play	作用： 播放指定音乐。 写法： Paly res/m1.wav
playloop	作用： 循环播放指定音乐。 写法： Play res/m1.wav
playstop	作用： 停止音乐的播放。 写法： Playstop 0 (0 为音乐索引号，按照添加顺序累加，直接 playstop 将停止所有音乐的播放)
petal	作用： 画面呈现樱花飞舞效果。 写法： Petal(停止为 petalstop)
snow	作用： 画面呈现银装素裹效果。 写法： Snow(停止为 snowstop)
rain	作用： 画面呈现落雨纷纷效果。 写法： Rain（停止为 rainstop）
set	作用： 预定义脚本变量。脚本变量也可在 AVGScreen 中通过 Command 类设置，也多用于 AVGScreen 与脚本交互，或者定义需要重复使用的数据信息。 写法：

	set var = “res/background.png” set var1 = 100

print	作用：
	将脚本中的预定义变量内容于指定位置显示出来。
	写法：
	1、set var = “res/background.png”
	mes print(var)
	2、set var =7
	If var == 7
	set inc=”res/my1.txt”
	else
	set inc=”res/my2.txt”
	endif
	include print(inc)
	1、 set var=10/2*36-7
	mes print(var)

reset	作用： 清空脚本中一切缓存，此时所有预定义变量都将不复存在。 写法： Reset
//,#.’	作用： 以上三者为行注释符号，自该符号起一整行数据将不被脚本读取。 写法： //XXXX #XXXX ‘XXXX
/**/	作用： 区域注释符号，自/*起，直到*/结束的所有数据将不被脚本读取。 写法： /* XXXXXXXXX XXXXX XXX */
exit	作用： 离开 AVGScreen 窗体。当执行此命令时脚本将被强制终止，并执行 onExit 函数，多用于转换 Screen。 写法： Exit

默认效果：

2、 org.loon.framework.android.game.action.avg.command

此包下共有类 5 个，分别为 Command.、CommandIterator、CommandType、Conversion、Expression 其中最主要者为 Command，此类为 AVG 引擎的核心解释器。

(1) Command

函数名	函数说明
initCommand()	静态函数，当此函数执行时，会初始 Command 中的所有缓存数据，此操作仅在缓存未实例化时有效。
formatCommand(String fileName)	格式化 Command 为指定脚本内容，此函数用以加载脚本到 Command 中。
formatCommand(String name, List resource)	格式化 Command 为指定脚本内容，此函数用以加载指定 List 命令集到 Command 中。
doExecute()	执行当前脚本行命令，并返回执行结果。
commandSplit(String src)	静态函数，用以拆分指定内容中赋值语句，并返回 List 格式。
openCache()	启动脚本缓存。
closeCache()	关闭脚本缓存。
nowCacheOffsetName()	获得当前缓存名称。
resetCache()	静态函数，执行后将清空所有脚本缓存。
setVariable(String key, Object value)	插入指定变量到脚本环境中。
setVariables(Map vars)	插入一组变量集合到脚本环境中。
removeVariable(String key)	删除指定名称的脚本变量。
Next()	此函数将返回一个布尔值，用以判断是否有下一行脚本存在。
gotoIndex(int index)	强制跳转向脚本的指定索引行。
batchToList()	批处理执行脚本，并将结果集以 List 形式返回。

batchToString()	批处理执行脚本，并将结果集以 String 形式返回。
getNameTags(……)	这是一系列的脚本标签分解方法，用以将文本信息转化为可用的脚本数据，并返回 List 数据集。

3、org.loon.framework.android.game.action.map

此包中有下属类 6 个，分别为 AStarFinder、Config、Field2D、TileList、TileMap、TileMapConfig，其中 AStarFinder 为 A*寻径使用、Config 中配置有基本的八方行走参数，Field2D 是 AStarFinder 使用的八方向寻径行走方向集，TileList 以及 TileMap 中提供有简单的游戏地图瓷砖操作功能。由于此部分并非核心功能，暂无详述，目前请以源码为主。

4、org.loon.framework.android.game.action.map.shapes

此包中下属有 7 种形状操作类，分别为 Circle、Triangle、RectBox、Point2D、Polygon2D、Vector2D 以及 Vector3D，可以满足一些常规的图形碰撞检测以及图形定向需要，由于 LGame 的 graphics.geom 包中照搬有 openJDK 的 geom 包，更复杂操作可以使用 geom 包来完成。由于此部分并非核心功能，暂无详述，目前请以源码为主。

5、org.loon.framework.android.game.action.sprite

此包下属类共 25 个，分别为 AbstractBackground、Animation、Background、Collidable、CollisionMask、ColorBackground、FlashLabel、GIFAnimation、IAnimation、ISprite、Label、Mask、Picture、Sprite、SpriteConfig、SpriteImage、SpriteMake、Sprites、SpriteTiled、StatusBar、StringComponent、WaitAnimation、SpriteRotate、ISpriteListener、SpriteListener、SpriteRotate。

其中核心类有六个，分别为 ISprite、Srpite、SpriteRotate、SpriteImage、SrpiteTiled、Sprites。

（1） ISprite

ISprite 是一个接口类，本身并不实现具体的业务，然而它却是所有精灵的基础，任何精灵都必须实现 ISprite 才能被 LGame 当作精灵类使用并有效管理。

函数名	使用方法
getWidth()	返回当前精灵的宽度。
getHeight()	返回当前精灵的高度。
getAlpha()	返回当前精灵的透明度。
x()	返回当前精灵的 x 轴坐标(int 形式)。
y()	返回当前精灵的 y 轴坐标(int 形式)。
getX()	返回当前精灵的 x 轴坐标(double 形式)。
getY()	返回当前精灵的 y 轴坐标(double 形式)。
setVisible(boolean visible)	设定当前精灵是否可见。
isVisible()	获得当前精灵是否可见 。
createUI(LGraphics g)	精灵底层绘制接口，可在此覆盖或修饰原有精灵画面。
update(long elapsedTime)	精灵用计时器，每当发生时间变化时此函数将被触发，也可以手动触发此函数引发精灵变动。
getLayer()	获得当前精灵所在层，层变动将直接影响精灵事件。

setLayer(int layer)	设定当前精灵所在层。
getCollisionBox()	返回当前精灵碰撞盒。

（2） Sprite

Sprite 是一个 ISprite 的具体实现，它内置有一套较为完整的精灵解决方案，如果不需要更为复杂的精灵实现时，直接使用此类即可满足绝大多数的精灵需求。

函数名	函数说明
setRunning(boolean runing)	设定是否允许精灵动画播放，当参数为 False 时动画播放将被暂停。
isRunning()	获得当前精灵动画是否被允许播放。
getTotalFrames()	获得当前精灵画面总帧数。
setCurrentFrameIndex()	强制将当前精灵画面转向指定帧。
getCurrentFrameIndex()	返回当前精灵画面所在帧。
setAnimation(……)	插入一组动画作为精灵播放。可以填写的参数为动画所在地址或图片(也可以直接插入一个动画类)，以及允许的最大帧数，分解动画时每幅小图宽，每幅小图高，以及平均每帧的播放时间。此方法相关函数较多，具体请参考实际开发时 IDE 提示。
getAnimation()	返回当前精灵所使用的动画。
update(long timer)	当精灵管理器发生时间变更或者手动触发此项事，将引发动画事件的变更。
updateLocation(Vector2D vector)	变更精灵到指定 vector 所在位置。
getImage()	返回当前帧的精灵图像（SpriteImage 格式）。
getLImage()	返回当前帧的精灵图像（LImage 格式）。
getImage(int index)	返回指定帧的精灵图像(LImage 格式)。
getWidth()	返回当前帧的精灵宽。
getHeight()	返回当前帧的精灵高。
getMiddlePoint()	获得当前精灵宽与高各除一半后的所在位置。
getDistance(Sprite second)	获得指定精灵与当前精灵间的中间差距数值。
getCollisionBox()	返回当前精灵的碰撞盒。
isRectToRect(Sprite sprite)	检查当前精灵与指定精灵之间是否发生了矩形碰撞。
isCircToCirc(Sprite sprite)	检查当前精灵与指定精灵之间是否发生了圆形碰撞。
isRectToCirc(Sprite sprite)	检查当前精灵与指定之间是否发生了矩形与圆形间的碰撞。
isPixelCollision(Sprite sprite)	检查当前精灵与指定精灵之间是否发生了像素级碰撞。
getMask()	获得当前精灵的碰撞掩码器，其中包含有本精灵当前帧的像素级信息。
createUI(LGraphics g)	绘制当前精灵当前帧到指定的 LGraphics。
setVisible(boolean visible)	设定当前精灵是否可见。
isVisible()	获得当前精灵是否可见。
setSpriteName(String	设定当前精灵名称。

spriteName)	
getSpriteName()	获得当前精灵名称。
setFilterType(int filterType)	设定当前精灵画面过滤方式。（int 参数来自 ImageFilterType 类，默认共 18 种滤镜可用）
getFilterType()	返回当前精灵画面过滤方式。
setAlpha(float alpha)	设定当前精灵透明度。
getAlpha()	返回当前精灵透明度。
setTransform(int transform)	设定当前精灵旋转角度。（int 参数来自 Trans 类，默认共 8 种角度可用）
getTransform()	返回当前精灵旋转角度。

（3） SpriteTiled

SpriteTiled 是 LGame 中使用的地图瓦片用类，用以构造我们需要的地图效果，以及获得地图与相关精灵的碰撞关系。

函数名	函数说明
setStaticTileSet(LImage image,int tileWidth, int tileHeight)	地图设定参数，该方法会将指定的 LImage 自动分解为我们要求大小的瓦片，并将此数据保存在 SpriteTiled 中，以供进一步操作使用。
createAnimatedTile(int staticTileIndex)	建立一个动态的瓦片分块，并与指定索引号关联。该方法会返回一个动态分块的索引号，索引由-1 开始分配。
setAnimatedTile(int animatedTileIndex,int staticTileIndex)	将索引号为参数一的动态分块，与参数二建立联系。
getAnimatedTile(int animatedTileIndex)	获得指定索引号的动态分块所关联的静态索引号。
setCell(int col, int row, int tileIndex)	在指定行、列位置的单元格设定分块索引。
getCell(int col, int row)	返回指定行、列位置的分块索引。
fillCells(int col, int row, int numCols, int numRows,int tileIndex)	用指定的分块索引设定一个区域中所有的单元格，该区域从 row 行、col 列起，累计包括 numRows 行以及 numCols 列。
getCellWidth()	返回当前精灵中分块宽度。
getCellHeight()	返回当前精灵中分块高度。
getColumns()	返回当前精灵中分块图层的单元格列数。
getRows()	返回当前精灵中分块图层的单元格行数。
createUI(LGraphics g)	SpriteTiled 绘图方法，用以绘制瓦片到指定 LGraphics 上，也可以重载此函数实现额外的图形绘制。
getTileAt(int x, int y)	返回指定 x，y 坐标对应的分块索引。
getWidth()	获得当前精灵实际宽度。
getHeight()	获得当前精灵实际高度。
update(long timer)	变更精灵动态事件。

setAlpha(float alpha)	设定当前精灵透明度。
getAlpha()	获得当前精灵透明度。
getCollisionBox()	获得当前精灵碰撞盒。
setVisible(boolean visible)	设定当前精灵是否可见。
isVisible()	获得当前精灵是否可见。

（4） Sprites

Sprites 是 LGame 使用的精灵管理器，它可以决定精灵的显示区域以及事件触发等细节因素，通常情况下 Sprites 需要和 Screen（或其子类）一起使用，另外所有的 Screen 中也都默认有一个直接可用的 Sprites 实例。

函数名	使用方法
sendToFront(ISprite sprite)	切换指定精灵到精灵管理器数组前方，此时该精灵将优先绘制。
sendToBack(ISprite sprite)	切换指定精灵到精灵管理器数组后方，此时该精灵将延后绘制。
sortSprites()	要求精灵管理器中的精灵再次排序，排序标准按照 Layer 的设定值。
add(int index, ISprite sprite)	添加一个精灵到精灵管理器中的指定索引位置。
add(ISprite sprite)	顺序添加精灵到精灵管理器中。
append(ISprite sprite)	顺序添加精灵到精灵管理器中(照顾 J2ME 用户习惯)。
getSprite(int index)	获得指定索引对应的精灵实例。
contains(ISprite sprite)	判断指定精灵是否存在于精灵管理器中。
remove(int index)	从精灵管理器中删除指定索引的精灵对象。
remove(ISprite sprite)	从精灵管理器中删除指定的精灵对象。
remove(int startIndex, int endIndex)	从精灵管理器中删除在指定索引范围内的精灵对象。
removeAll()	从精灵管理器中删除全部的精灵对象。
clear()	从精灵管理器中 NULL 化全部的精灵对象。
update(long elapsedTime)	触发精灵管理器的时间变更方法，此时管理器中所有精灵的同名方法也会被一起触发。
createUI(LGraphics g)	绘制精灵管理器中图像到指定 LGraphics。
createUI(LGraphics g, int x, int y)	绘制精灵管理器中图像到指定 LGraphics 的指定坐标。
setViewWindow(int x, int y, int width, int height)	设定精灵管理器的显示位置与显示大小（可以实现简单的屏幕移动）。
public void setLocation(int x, int y)	设定精灵管理器的显示位置。
getSprites()	返回当前精灵管理器中的全部精灵对象。
size()	获得当前精灵管理器中的精灵总数。
getWidth()	获得当前精灵管理器所允许的可显示宽度。
getHeight()	获得当前精灵管理器所允许的可显示高度。

setVisible(boolean visible)	设定当前精灵管理器是否可见。
isVisible()	判断当前精灵管理器是否可见。

PS：此处需要注意一点，在 LGame 中纯绘制的动态可控组件共分为精灵与组件(或者说Window)两种。出于效率考虑，默认情况下 LGame 精灵无法直接获得触摸屏或键盘事件（需要调用相关函数判定碰撞信息）。如果想要直接获得相关物体是否被点击的信息，可以使用组件完成（譬如让 LPaper 插入动画，而后重载其 onClick 函数）。从具体实现上讲，组件除拥有更为复杂（也更为耗时）的自身状态判定外，与精灵是完全对等的存在（大多数情况下，两者的具体实现也都继承成自 LObject）。

6、org.loon.framework.android.game.action.sprite.effect

此包中共有类 6 个，分别为 Fade、FreedomEffect、IKernel、PetalKernel、RainKernel、SnowKernel，他们为 LGame 提供了一些简单的动画效果，由于并非核心功能，所以目前不详加介绍。值得注意的是，LGame 中的特效类本身也是精灵，可以直接按照精灵类使用。

7、org.loon.framework.android.game.action.sprite.j2me

此包中共有类 7 个，分别为 Command、CommandListener、J2MEKey、Layer、LayerManager、Sprite、TiledLayer，他们为 LGame 提供了一组仿 J2ME 游戏精灵的功能，作用在于帮助不熟悉 LGame 开发，但拥有 J2ME 使用经验的用户也能轻松使用 LGame 或者快速移植 J2ME 游戏到 Android 中。

除 J2ME 中原有的 Image 图形部分，需要使用 LGame 提供的 LImage（100%兼容 Image API）外，此包中类与方法同 J2ME 中同名类操作完全一致，故不再赘述，具体请参看 J2ME 相关部分文档。值得一提的是，虽然 LGame 并没有硬性规定此包及其下属类的使用场合，但通常与 LGame 中高仿 GameCanvas 实现的 CanvasScreen 类一起使用效果较好。

8、org.loon.framework.android.game.core

此包中共有类 11 个，分别为 EmulatorButton、EmulatorButtons、EmulatorListener、GIFDecoder、LGUID、LInput、LObject、LSystem、LUUID、Runner、Version，自 core 包开始，会涉及到 LGame 中最基础的业务逻辑，此包中最核心的两个类是 LSystem 以及 LObject。

(1) LSystem

LSystem 是 LGame 引擎的核心类之一，其中承载有大量的引擎必备静态函数，假如删除此类，那么整个 LGame 都将无法运作。

函数名	函数说明
getOSVersion()	获得当前手机系统版本。
getSDKVersion()	获得当前手机 SDK 名称。
getMajorOSVersion()	获得当前手机系统版本标识（0 为 1.1 版、1 为 1.5 版、2 为 1.6 版、3 为 2.0 版、4 为 2.1 版、5 为 2.2 版）
isEmulator()	获得当前系统是否正在使用模拟器。
isSamsungGalaxy()	获得当前系统是否为三星手机。
isDroidOrMilestone()	获得当前系统是否为摩托罗拉手机。

destroy()	注销 LGame 引擎中的常规缓存资源。
getSurface2D()	获得当前系统的 2D 图形处理接口（内部封装有 LGameAndroid2DView 的图像相关部分）
exit()	强制退出 LGame 引擎系统。
repaint()	强制刷新手机画面。
repaint(Bitmap bit)	强制刷新手机画面为指定 Bitmap。
repaint(LImage img)	强制刷新手机画面为指定 LImage。
repaint(Bitmap bit, int w, int h)	强制刷新手机画面为指定 Bitmap 并按照 w,h 参数的比例居中显示。
repaint(LImage img, int w, int h)	强制刷新手机画面为指定 LImage 并按照 w,h 参数的比例居中显示。
repaintFull(Bitmap bit, int w, int h)	强制扩大 Bitmap 为指定 w,h 并按比例居中显示于手机画面之上。
repaintFull(LImage img, int w, int h)	强制扩大 LImage 为指定 w,h 并按比例居中显示于手机画面之上。
repaintResize(Bitmap bit, int w, int h)	强制扩大或缩小 Bitmap 为指定 w,h 并按比例居中显示于手机画面之上。
repaintResize(LImage img, int w, int h)	强制扩大或缩小 LImage 为指定 w,h 并按比例居中显示于手机画面之上。
repaintLocation(Bitmap bit, int x, int y)	强制刷新指定 Bitmap 至手机画面的指定位置。
repaintLocation(LImage img, int x, int y)	强制刷新指定 LImage 至手机画面的指定位置。
setFPS(long fps)	设定当前画面刷新速度。（仅对标准 Screen 有效）
getFPS()	获得当前画面实际刷新速度。（仅对标准 Screen 有效）
getMaxFPS()	获得当前画面所允许的最大刷新速度。（仅对标准 Screen 有效）
getActivity()	获得当前游戏所使用的 Activity。
getScreenOrientation()	获得当前屏幕方向的字符串形式。
getScreenOrientation(Activity activity)	获得指定 Activity 屏幕的字符串形式。
looksLikePortrait(Activity activity)	获得指定 Activity 屏幕是否为纵屏。
looksLikeLandscape(Activity activity)	获得指定 Activity 屏幕是否为横屏。
looksLikeSquare(final Activity activity)	获得指定 Activity 屏幕是否为正方屏（宽高对等）。
getRandom(int i, int j)	获得一个指定范围内的随机数。
getSystemTimer()	获得当前 LGame 所使用的默认计时器实例。
setupHandler(LGameAndroid2DActivity activity,LGameAndroid2DView view, boolean landscape)	为 LGame 设定游戏必须的 Activity、View 以及屏幕方向。

getVersionName()	返回当前游戏版本名称。
getCacheFile()	返回当前游戏缓存文件夹的 File 形式。
getCacheFile(String fileName)	返回当前游戏缓存文件夹下指定文件的 File 形式。
getCacheFileName()	返回当前游戏缓存文件夹的路径字符串。
getSystemHandler()	获得当前游戏句柄。（内含 Activity、View 等的封装）
gc()	强制调用 GC 命令。
gc(int size, long rand)	以指定范围中的指定随机率调用 GC 命令。
gc(long rand)	以百分比的随机率调用 GC 命令。(如 LSystem.gc(10);为 10%的可能性调用 GC 命令）
read(byte[] buffer)	将指定比特数组转化为数据流。
InputStream read(File file)	将指定 File 文件转化为数据流。
InputStream read(String fileName)	将指定路径文件转化为数据流。
loadPropertiesFileToSystem (final File file)	将指定 File 文件导入系统默认的 Properties 配置。
loadPropertiesFromFile(final File file)	将指定 File 文件导入 Properties 并返回获得的 Properties。
loadProperties(Properties properties, InputStream inputStream, String fileName)	把指定的输入数据流导入指定的 Properties 当中。
writeInt(OutputStream out, int number)	在指定输出流中写入一个整型数字。
readInt(InputStream in)	从指定输入流中获得一个整型数字。

(2) LObject

LObject 是一个特殊的抽象类，事实上可以将他认为是 LGame 中组件与精灵的共同父类，在他内部封装了一些组件与精灵的共同操作，只要继承自他的组件或精灵都可以使用这些操作。

函数名	使用方法
setLayer(int layer)	设定目前对象所在层。
getLayer()	获得目前对象所在层。
move_45D_up()	命令当前对象往 45 度角（八方向）的上方运动一步。
move_45D_up(int multiples)	命令当前对象往 45 度角（八方向）的上方以指定速度倍数运动一步。
move_45D_left()	命令当前对象往 45 度角（八方向）的左方运动一步。
move_45D_left(int multiples)	命令当前对象往 45 度角（八方向）的左方以指定速度倍数运动一步。
move_45D_right ()	命令当前对象往 45 度角（八方向）的左方运动一步。

move_45D_right (int multiples)	命令当前对象往 45 度角（八方向）的右方以指定速度倍数运动一步。
move_45D_down ()	命令当前对象往 45 度角（八方向）的下方运动一步。
move_45D_down (int multiples)	命令当前对象往 45 度角（八方向）的下方以指定速度倍数运动一步。
move_up()	命令当前对象向下移动一步。
move_up(int multiples)	命令当前对象以指定速度倍数向下移动一步。
move_left ()	命令当前对象向左移动一步。
move_left (int multiples)	命令当前对象以指定速度倍数向左移动一步。
move_right ()	命令当前对象向右移动一步。
move_right(int multiples)	命令当前对象以指定速度倍数向右移动一步。
move_down ()	命令当前对象向下移动一步。
move_down(int multiples)	命令当前对象以指定速度倍数向下移动一步。
move(Vector2D vector2D)	命令当前对象按照指定 vector 移动。
move(double x, double y)	命令当前对象移动到指定位置。
setLocation(double x, double y)	设定当前对象所在坐标。
x()	返回当前对象 x 坐标。（int 形式）
y()	返回当前对象 y 坐标。（int 形式）
getX()	返回当前对象 x 坐标。（double 形式）
getY()	返回当前对象 y 坐标。（double 形式）
setX(……)	设定当前对象 x 坐标。
setY(……)	设定当前对象 y 坐标。
getLocation()	获得当前对象的 Vector2D 形状。
setLocation(Vector2D location)	设定当前对象的 Vector2D 形状。
getWidth()	返回当前对象宽。
getHeight()	返回当前对象高。
update(long timer)	触发当前对象时间点变更。
createUI(LGraphics g)	绘制当前对象到指定的 LGraphics 之上。

9、org.loon.framework.android.game.core.graphics

该包下共有类 13 个，分别为 CanvasScreen、Desktop、Draw、DrawImpl、IScreen、LColor、LComponent、LContainer、LFont、LImage、Screen、ThreadScreen、Trans，其中最为重要的类为 Screen、ThreadScreen、CanvasScreen、LComponent、Desktop、LImage。另外 LFont 和 LColor 是完全参照 Java 中名称近似类 API 稍有简化而成的。

（1） Screen

Screen 继承自 IScreen，是 LGame 引擎提供的四种 Screen 之一，也是其它三种 Screen 的父类，与其它三种 Screen 一样，它的最主要作用就在于显示及控制游戏画面到手机当中。另外，这个默认的 Screen 采取自动刷新模式，没有手动的 repaint 函数，刷新速度由 setFPS 函数决定。

函数名	函数说明
onLoad()	该函数运行在一个独立的线程中，只会在 Screen 构建完毕，并且 setScreen 到游戏中时运行一次，可以在其中放置一些较为耗时的初始赋值操作。
getName()	返回当前 Screen 的实际名称。
onDirection(SensorDirection direction, float x, float y, float z)	当 LGame 调用 setupGravity 时生效，重载此函数即可获得当前的手机感应方向与针对手机陀螺仪的 x，y，z 三点坐标。（需要注意的是，LGame 为防止方向混淆，仅为竖屏状态提供了上、下、左、右四方向判定，而横屏时只能自动判定手机的左右摇摆）。
onResume()	当手机重新进入游戏系统时会调用此函数，重载即可使用。
onPause()	当手机暂时离开游戏系统时会调用此函数，重载即可使用。
onCreateOptionsMenu(Menu menu)	当按下手机的 menu 按键时，此函数生效，将根据设置为手机创建菜单。
onOptionsItemSelected(MenuItem item)	当已按下 menu 按键，并且菜单处于正常显示状态时，此函数生效，可以在此获得具体的菜单选中项并触发对应事件。
onOptionsMenuClosed(Menu menu)	当已按下 menu 按钮的前提下，再次按下 menu，或者人工 close 菜单时生效，此函数将在菜单结束前被触发，可以在此时进行一些收尾工作。
setEmulatorListener(EmulatorListener emulator)	设定当前 Screen 使用的模拟按钮监听器，当此项不为 NULL 时，LGame 会自动绘制一组支持多点触摸的模拟按钮到游戏中去。另外，一旦 Screen 实现了此监听器，也会达到相同的效果。
getEmulatorButtons()	返回当前 Screen 使用的模拟按钮集合。
emulatorButtonsVisible()	设定模拟按钮集合是否可见。
openBrowser(String url)	要求 Android 浏览指定的网页内容。
isVisibleAD()	获得当前 Admob 广告是否可见。
hideAD()	隐藏 Admob 广告。
showAD()	显示 Admob 广告。
showAndroidProgress(String title,String message)	弹出一个指定内容的进度条。
showAndroidAlert(String title,String message)	弹出一个指定内容的提示框。
showAndroidSelectMessageBox(String title, String text[])	弹出一个指定内容的选择框。
showAndroidExceptionMessageBox(Exception ex)	弹出一个指定内容的异常信息。
getAndroidSelect()	返回打开的选择框选中项。
playtAssetsMusic(String file, boolean loop)	播放 Assets 文件夹下的指定音频文件（当前 Android 音频播放器默认不支持数据流，只能放于此处），当布尔值为 True 时循环播放，否则为 False 仅播放一次。
resetAssetsMusic(int vol)	设定当前音频文件的播放音量。
resetAssetsMusic()	刷新当前音频文件的播放设置。

stopAssetsMusic()	停止当前音频文件的播放。
stopAssetsMusic(int index)	停止指定索引的音频文件播放。
setBackground(……)	设定当前 Screen 的背景图片。（可以为 LImage 或 Bitmap 格式）
setBackground(String fileName,boolean transparency)	设定指定路径文件为当前 Screen 的背景图片。当 transparency 为 False 时，尝试以 RGB565 模式加载，当 transparency 为 True 时，尝试以 ARGB4444 模式加载。
setBackground(LColor color)	设定当前 Screen 背景为指定颜色。
getBackground()	返回当前 Screen 的背景图片。（Bitmap 格式）
setFPS(long fps)	设定当前 Screen 画面刷新速度。
getFPS()	返回当前 Screen 画面实际刷新速度。
destroy()	强制注销当前 Screen 窗体。
dispose()	当切换不同 Screen，或者当前 Screen 时会触发此方法，此时可以针对目前 Screen 进行一些诸如资源释放的善后工作。
getDesktop()	获得当前 Screen 的组件管理器。
getSprites()	获得当前 Screen 的精灵管理器。
add(……)	从 Screen 中添加一个指定的精灵或组件。
remove(……)	从 Screen 中删除一个指定的精灵或组件。
removeAll()	删除 Screen 中的全部精灵和组件。
onClick(……)	判断是否点中了指定的精灵或者组件。
centerOn(LObject object)	居中指定精灵或组件。
topOn(LObject object)	置顶指定精灵或组件。
leftOn(LObject object)	将指定精灵或组件移向最左方。
rightOn(LObject object)	将指定精灵或组件移向最右方。
bottomOn(LObject object)	将指定精灵或组件移向画面最底。
setShakeNumber(int shake)	设定画面的振动频率，当此数值非-1 且非 0 时画面会产生振动。
getShakeNumber()	获得当前画面的振动数值。
callEvent(Runnable runnable)	运行指定的线程。
callEventWait(Runnable runnable)	暂停指定的线程。
callEventInterrupt()	中断所有在 callEvent 使用过且尚未完成的线程。
callEvents()	运行所有注入 callEvent 的线程事件。
callEvents(boolean execute)	运行所有注入 callEvent 的线程事件，当布尔值为 False 时则清空所有注入的线程事件。
getWebImage(String url, boolean transparency)	从指定网络路径获得 LImage 资源。当 transparency 为 False 时，尝试以 RGB565 模式加载，当 transparency 为 True 时，尝试以 ARGB4444 模式加载。
getImage(String fileName, boolean transparency)	从指定本地路径获得 LImage 资源。当 transparency 为 False 时，尝试以 RGB565 模式加载，当 transparency 为 True 时，尝试以 ARGB4444 模式加载。
getSplitImages(String fileName, int row, int	尝试分解指定路径的 LImage 资源为指定大小的小图并返回 LImage[]。当 transparency 为 False 时，尝试以 RGB565 模式加

col,boolean transparency)	载，当 transparency 为 True 时，尝试以 ARGB4444 模式加载。
getSplitImages(LImage image, int row, int col)	尝试分解指定的 LImage 资源为指定大小的小图并返回 LImage[]。当 transparency 为 False 时，尝试以 RGB565 模式加载，当 transparency 为 True 时，尝试以 ARGB4444 模式加载。
createUI(LGraphics g)	绘制 Screen 画面到指定 LGraphics 之上，此方法会在 setScreen 将 Screen 注入指定 View 后自动调用。
draw(LGraphics g)	抽象函数，绘制 Screen 画面到指定 LGraphics 之上，此函数默认的 LGraphics 自动获取于手机 View。
runTimer(LTimerContext timer)	运行 Screen 的时间点变更事件，此函数将直接影响到所有 Screen 下属的组件及精灵 update 方法。
setNext(boolean next)	设定是否允许当前画面刷新。
next()	返回是否允许当前画面刷新。
alter(LTimerContext timer)	游戏资源变更用函数，每当 Screen 时间点发生变化时，都会触发此函数执行。
setScreen(IScreen screen)	为游戏切换一个新的 Screen，只要是实现自 IScreen 的窗体都可以于此处注入。
getWidth()	获得 Screen 宽度。
getHeight()	获得 Screen 高度。
refresh()	刷新 Screen 触摸屏及键盘参数。
pause(long timeMillis)	暂停 Screen 线程为指定时间。
getTouch()	获得当前 Screen 触摸点的 Point 格式。
isPaused()	判定手机是否触发了暂停事件。
isClick()	判断当前 Screen 是否被人点击。
isTouchClick()	判断当前 Screen 是否被人点击（IScreen 中无需实现此方法，但 LInput 要求实现）。
isTouchClickUp()	判断当前 Screen 是否被人释放点击。
getTouchPressed()	获得当前 Screen 点击状态参数。
getTouchReleased()	获得当前 Screen 点击释放状态参数。
isTouchPressed(int button)	获得当前 Screen 是否点击并触发了指定索引的事件。
isTouchReleased(int button)	获得当前 Screen 是否释放点击并触发了指定索引的事件。
getKeyDown(int keyCode)	获得当前 Screen 是否执行了指定的键盘事件。
getKeyPressed(int keyCode)	获得当前 Screen 是否释放了指定的键盘事件。
getTouchX()	获得在触摸屏上点击的 X 坐标。
getTouchY()	获得在触摸屏上点击的 Y 坐标。
getTouchDX()	获得在触摸屏上拖动经过的 X 坐标。
getTouchDY()	获得在触摸屏上拖动经过的 Y 坐标。
isTouchDown(int button)	判断指定的触摸屏事件是否得到了执行。
isKeyDown (int keyCode)	判断指定的键盘事件是否得到了执行。
onKeyDown(int keyCode, KeyEvent e)	触发键盘按下事件时将调用此函数。
onKeyUp(int keyCode, KeyEvent e)	触发键盘放开事件时将调用此函数。
onTouchMove(MotionEvent	触发触摸屏移动（拖拽）事件时将调用此函数。

e)	
onTouchDown(MotionEvent e)	触发触摸屏按下事件时将调用此函数。
onTouchUp(MotionEvent e)	触发触摸屏放开事件时将调用此函数。
onTouch(float x, float y, MotionEvent e, int pointerCount, int pointerId)	仅在支持多点触摸的环境中有效，触发多点触摸时将调用此函数。
move(double x, double y)	强制变更触摸点为指定 x,y。
isMoving()	判断目前是否正在移动（拖拽）触摸屏。
getHalfWidth()	获得当前 Screen 的一半宽。
getHalfHeight()	获得当前 Screen 的一半高。
getTouchDirection()	获得当前 Screen 中的触摸屏点击方向。（返回的 int 数值对应于 LInput 中的相关参数）

（2） ThreadScreen

ThreadScreen 是 Screen 的子类，它的特点在于拥有独立的内部线程，除了常规的 LGraphics 绘图接口外还拥有一个更为简化的 Draw 绘图接口，在 0.2.9 版中它被调整为自动刷新画面，不过也可以使用 setRunning(false)方式来停止自动刷新（即停止自动刷新的线程）。

函数名	函数说明
text(String s, int x, int y)	在窗体指定位置绘制文字。
text(int x, int y, String s)	在窗体指定位置绘制文字(文字有加粗边框)。
text(String s, int y)	在窗体 x 坐标居中的前提下，在指定 y 坐标上绘制文字。
draw(LImage img, int x, int y)	绘制 LImage 到指定 x,y 坐标。
color(int r, int g, int b)	设定当前窗体的 rgb 颜色。
rect(int x, int y, int w, int h)	设定当前窗体的绘图区域。
fill(int x, int y, int w, int h)	填充当前窗体的绘图区域。
resizeScreen(int w, int h)	变更当前窗体中画布为指定大小。
loadSound(String fileName)	加载指定路径的音频文件到当前窗体。
getBenchCount()	获得当前线程运行的时间长度。
sleep(long i)	让窗体暂停指定时间。（暂停方式为 while 循环阻断）
getSleepTime()	获得当前窗体的暂停时间。
setSynchroFPS(int fps)	设定针对当前窗体的 FPS。
getSynchroFPS()	返回针对当前窗体设定的 FPS。
getDraw()	返回一个诞生自 LGraphics 的 Draw 绘图器。
getImage()	返回针对当前窗体的 LImage。
getLGraphics()	返回针对当前窗体的 LGraphics。
initialize()	此为抽象函数，仅在此类及其子类初始化结束后会被调用。
drawScreen(LGraphics g)	此为抽象函数，用以绘制游戏画面到手机当中。
repaint()	刷新游戏画面。

gameLoop()	此为抽象函数，当游戏每次循环时都会自动调用此函数。
setSynchro(boolean synchro)	设定窗体是否启动 FPS 同步。
isSynchro()	判定窗体是否启动了 FPS 同步。
setRunning(boolean isRunning)	设定游戏线程是否运行。
isRunning()	判定游戏线程是否运行。
setRunPriority(int runPriority)	设定当前窗体的线程级别。
getRunPriority()	获得当前窗体的线程级别。
getExtWidth()	获得变更画布大小后的窗体画布宽度。
getExtHeight()	获得变更画布大小后的窗体画布高度。
setCenterX(int centerX)	强制设定当前窗体画布在手机上的 X 坐标。
setCenterY(int centerY)	强制设定当前窗体画布在手机上的 Y 坐标。

（3） CanvasScreen

CanvasScreen 继承自 Screen，它是一个高仿 J2ME 中 GameCanvas 类的特殊窗体，仅在执行 repaint 时才会刷新画面。

函数名	函数说明
setSize(int w,int h)	变更窗体中画布为指定大小。
getKeyStates()	返回当前键盘状态。
getCurrentWidth()	获得窗体中实际的画布宽度。
getCurrentHeight()	获得窗体中实际的画布高度。
getGameAction(int keyCode)	转换 Android 键盘事件为对应的 J2ME 游戏按键标识并返回。
getKeyCode(int gameAction)	转换 J2ME 游戏按键标识为对应的 Android 键盘事件并返回。
flushGraphics(int x, int y, int width, int height)	刷新指定范围的游戏画布。
flushGraphics()	刷新游戏画布。
getGraphics()	获得当前窗体的 LGraphics。
setFullScreenMode(boolean full)	设定游戏画布是否全屏。
isFullScreenMode()	判定游戏画布是否全屏。
repaint()	刷新游戏画布。
repaint(int x, int y, int width, int height)	刷新指定范围的游戏画布。
paint(LGraphics g)	抽象函数，用以绘制游戏画面到手机当中。
pointerPressed(double x, double y)	点击触摸屏。

pointerMove(double x, double y)	移动触摸屏。
pointerReleased(double x, double y)	释放触摸屏。
keyPressed(int keyCode)	键盘按下。
keyReleased(int keyCode)	键盘放开。
exitGame()	离开游戏。

（4） LComponent

LComponent 是 LGame 引擎提供的纯绘制基础组件，也是 LContainer 的父类，它和 LContainer 的最大区别在于不能直接在 LComponent 上 add 一个新组件让它们一起联合行动（LContainer 可以），在作用上 LComponent 有些近似于精灵类的 ISprite。

函数名	函数说明
isContainer()	判断当前是“容器”还是“组件”，在未被 LContainer 继承的前提下，此项将显示为 False，也就是非容器。
update(long timer)	更新组件时间点。
createUI(LGraphics g)	绘制当前组件画面到指定的 LGraphics 之上。
createCustomUI(LGraphics g, int x, int y, int w, int h)	这是一个友元函数，用以扩展用户需要的画面到组件之上。
intersects(int x, int y)	检查当前组件与指定坐标是否有交叉。
intersects(LComponent comp)	检查当前组件与指定组件是否有交叉。
dispose()	释放当前组件资源。
setVisible(boolean visible)	设定当前组件是否显示。
isVisible()	返回当前组件是否显示。
setEnabled(boolean b)	设定当前组件是否可用。
isEnabled()	返回当前组件是否可用。
setSelected(boolean b)	设定当前组件是否被选中。
isSelected()	返回当前组件是否被选中。
requestFocus()	转移组件管理器中组件焦点到当前组件。
transferFocus()	转让组件管理器中当前组件焦点。
transferFocusBackward()	转让组件管理器中当前组件焦点，并将当前组件放置于组件管理器最后。
setFocusable(boolean b)	设定焦点。
isFocusable()	判断是否为焦点。
setContainer(LContainer container)	设定容器。
getContainer()	返回组件中容器。
setDesktop(Desktop desktop)	设定组件所属的组件管理器。
setBounds(double dx,	设定组件形状。

double dy, int width, int height)	
setLocation(double dx, double dy)	设定组件坐标。
move(double dx, double dy)	命令组件朝指定坐标移动。
setSize(int w, int h)	变更组件大小设置。
setWidth(int width)	设定组件宽度。
setHeight(int height)	设定组件高度。
getWidth()	返回组件宽度。
getHeight()	返回组件高度。
getScreenX()	返回组件在 Screen 中的 x 坐标。
getScreenY()	返回组件在 Screen 中的 y 坐标。
getCollisionBox()	返回组件碰撞盒。
setAlpha(float alpha)	设定组件透明度。
getAlpha()	返回组件透明度。
setBackground(……)	设定组件背景图片，可以分别注入 LColor、LImage 或者图片路径，当 t 为 False 时，尝试以 RGB565 模式加载，当 t 为 True 时，尝试以 ARGB4444 模式加载。
getBackground()	获得当前组件背景图片。
setImageUI(……)	设定当前组件所使用的前景 UI 图片。
getImageUI()	返回当前组件所使用的前景 UI 图片。
getUIName()	返回当前组件实际名称。

（5） Desktop

Desktop 是 LGame 引擎所使用的组件资源管理器，它的实际作用相当于精灵类中的 Sprites，比较特殊的是 Desktop 管理组件依靠 LContainer（容器类，也是 LComponent 的子类）。

函数名	函数说明
add(LComponent comp)	添加一个组件到资源管理中。
remove(LComponent comp)	从资源管理器中删除一个指定的组件。
update(long timer)	组件资源管理器时间点更新。
createUI(LGraphics g)	绘制组件资源管理器到指定的 LGraphics 当中。
clearFocus()	清除资源管理器中所有组件的焦点。
validateUI ()	重新验证当前资源管理器中所有 UI。（当组件有变化时将会执行变化）
setSize(int w, int h)	变更资源管理器大小。
getWidth()	返回资源管理器宽度。
getHeight()	返回资源管理器高度。
setContentPane(LContainer pane)	设定当前资源管理器所用的基础容器。
getContentPane()	返回当前资源管理器所使用的基础容器。

installUI(UIConfig newConfig)	安装指定的组件配置集合。
getUIConfig()	获得当前所使用的组件配置集合。
get()	返回当前资源管理器的组件形式。

（6） LImage

LImage 是 LGame 引擎特有的图像资源类，内部为 Bitmap 封装，其 API 兼顾了 J2SE 及 J2ME 版的 Java Image 类(偏向于 J2ME)。需要特别注意的是，由于 Android 环境图形资源有限，当某个 LImage 不再使用时，请一定要手动调用 dispose()释放相关资源，以减少不必要的资源损耗。

函数名	函数说明
createImage(InputStream in, boolean transparency)	从指定输入流转化出 LImage 图形，并尝试根据 transparency 将其设定为 RGB565（False 时）或 ARGB4444（True 时）格式。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。
createImage(byte[] buffer)	将指定的比特数组转换为 LImage 图形。
createImage(byte[] buffer, boolean transparency)	将指定的比特数组转换为 LImage 图形，并尝试根据 transparency 将其设定为 RGB565（False 时）或 ARGB4444（True 时）格式。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。
createImage(int width, int height, boolean transparency)	创建一张指定大小的 LImage 图像，并根据 transparency 将其设定为 RGB565（False 时）或 ARGB4444（True 时）格式。(此设定将绝对生成指定的图像格式)
createImage(int width, int height)	创建一张指定大小的 LImage 图像。
createImage(int width, int height, Config config)	创建一张指定大小的 LImage 图像，并且将其设定为 Config 所指定的图像格式。
createImage(byte[] imageData, int imageOffset, int imageLength, boolean transparency)	从指定比特数组中，截取指定长度数据生成 LImage 图像，并尝试根据 transparency 将其设定为 RGB565（False 时）或 ARGB4444（True 时）格式。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。
createImage(byte[] imageData, int imageOffset, int imageLength)	从指定比特数组中，截取指定长度数据生成 LImage 图像。
createImage(String fileName)	加载 APK 中指定路径资源为 LImage。（有缓存）

createRGBImage(int[] rgb, int width, int height, boolean processAlpha)	根据指定像素集合，创建一张指定大小的 LImage 图像，当 processAlpha 为 True 时，图像格式为 ARGB4444，否则为 RGB565。
createImage(LImage image, int x, int y, int width, int height, int transform)	从指定的 LImage 中指定坐标起始，截取指定宽度及高度的图片，并且根据 transform 参数旋转为之指定角度（相关 int 参数来自于 Trans 中静态常量），并返回新生的 LImage 图像。
createImage(int count, int w, int h,boolean transparency)	创建出指定数量、指定大小的 LImage 图像，并根据 transparency 将其设定为 RGB565（False 时）或 ARGB4444（True 时）格式。
createImage(int count, int w, int h, Config config)	创建出指定数量、指定大小的 LImage 图像，并根据 Config 设置将其生成为指定格式。
setBitmap(Bitmap bitmap)	重新设置 LImage 的内部 Bitmap。
getBitmap()	返回 LImage 的内部 Bitmap。
getConfig()	获得对应当前图像资源的 Config 配置。值得注意的是，Android 系统并不总能获得正确的图像格式。在 Bitmap 中，当遇到无法识别的格式时它将被返回为 NULL，而 LImage 则强行将它们归结为 ARGB8888 格式（一种非常清晰，也非常消耗资源的格式）。
clone()	依据现有 LImage，克隆出另外一个 LImage。
getLGraphics()	返回当前 LImage 的 Graphics 实例。（LGame 引擎特有的 LGraphics）
create()	无视是否有打开的 Graphics 实例，重新生成一个 LGraphics 实例。
getWidth()	获得当前图像宽度。
getHeight()	获得当前图像高度。
getPixels()	返回当前图像的像素集合。
getPixels(int pixels[])	将当前图像的像素集合注入指定整型数组并返回。
getPixels(int x, int y, int w, int h)	返回指定范围内的当前图像像素集合。
getPixels(int offset, int stride, int x, int y, int w, int h)	返回指定范围内的当前图像像素集合。
getPixels(int pixels[], int offset, int stride, int x, int y, int width, int height)	返回指定范围内的当前图像像素集合到指定的整型数组中去。
setPixels(int[] pixels, int w, int h)	将当前 LImage 中图像资源设定为指定范围内的像素集合。
setPixels(int[] pixels, int offset, int stride, int x, int y, int width, int height)	将当前 LImage 中图像资源设定为指定范围内的像素集合。
setPixels(int[] pixels, int x, int y, int w, int h)	将当前 LImage 中图像资源设定为指定范围内的像素集合。
getPixel(int x, int y)	返回指定坐标对应的像素点数据。
getRGB(int pixels[], int offset, int stride, int x, int y, int width, int height)	将当前 LImage 中图像资源设定为指定范围内的像素集合。(照顾 Java 同名函数习惯)
getRGB(int x, int y)	返回指定坐标对应的像素点数据。(照顾 Java 同名函数习惯)

setPixel(LColor c, int x, int y)	设定指定 LColor 像素点到当前图形的指定坐标之上。
setPixel(int rgb, int x, int y)	设定指定像素点到当前图形的指定坐标之上。
setRGB(int rgb, int x, int y)	设定指定像素点到当前图形的指定坐标之上。(照顾 Java 同名函数习惯)
convertConfig(Config config)	强制转换 LImage 为指定图像格式。
getCacheSubImage(int x, int y, int w, int h)	从当前图像截取出指定范围内的小图。（有缓存）
getCacheSubImage(int x, int y, int w, int h,boolean transparency)	从当前图像截取出指定范围内的小图，并根据 transparency 将其设定为 RGB565（False 时）或 ARGB4444（True 时）格式。（有缓存）
getSubImage(int x, int y, int w, int h)	从当前图像截取出指定范围内的小图。（无缓存）
getSubImage(int x, int y, int w, int h, boolean transparency)	从当前图像截取出指定范围内的小图，并根据 transparency 将其设定为 RGB565（False 时）或 ARGB4444（True 时）格式。（无缓存）
scaledInstance(int w, int h)	返回一个指定大小的当前图像资源。
hashCode()	获得当前图像的哈希序列。（此为依据图像像素点，采取快速截取法生成的 hash，有可能在近似图中存在误差）
isClose()	判断当前 LImage 是否已经被关闭。
dispose()	释放当前 LImage 资源。（主要为 recycle 内部的 Bitmap）

10、 org.loon.framework.android.game.core.graphics.device

在此包中仅含有 2 类存在，却都非常的重要，他们一个是照搬自 LGame-J2SE 版，融合 J2SE+J2ME 图形 API 并有所扩充的 LGraphics 接口；一个是封装自 Canvas，也是作为 LGraphics 接口具体实现的 LGraphicsAndroid2D。在 LGame 引擎中，Android2D 部分的所有图形绘制都依赖于 LGraphics。

函数名	函数说明
initGraphics()	初始化游戏画布，当此函数调用后，所有关于 LGraphics 的参数都会被初始化，即使该 LGraphics 已经被关闭，也将重新获得启动。
reset()	刷新游戏画布，此函数可以重新部署 LGraphics 的设置参数，但是仅在 LGraphics 被关闭前有效。
getMatrix()	获得一组 Matrix 信息。
getInverseMatrix()	获得一组左右颠倒的 Matrix 信息。
create()	创建一个新的 LGraphics 对象，它是当前 LGraphics 对象的克隆。
create(int x, int y, int w, int h)	基于此 LGraphics 创建一个新的 LGraphics 对象，但是使用新的转换和剪贴区域。
setAntiAlias(boolean flag)	设定当前图形绘制模式为优秀或普通。
isAntiAlias()	获得当前图形绘制模式为优秀或普通。
setAlphaValue(int alpha)	设定当前图形透明度。（0-255）

setAlpha(float alpha)	设定当前图形透明度。(0.0F-1.0F)
getAlphaValue()	返回当前图形透明度。（0-255）
getAlpha()	返回当前图形透明度。（0.0F-1.0F）
setColor(……)	设定当前图形使用的色彩，可以为 RGB、ARGB 数值或者直接使用 LColor 注入。
getColor()	返回当前图形所使用的色彩。（LColor 格式）
getColorRGB()	返回当前图形所使用的色彩为 RGB 模式。
getColorARGB()	返回当前图形所使用的色彩为 ARGB 模式。
setGrayScale(int grey)	设置绘图所使用的颜色灰度。
getGrayScale()	返回绘图所使用的颜色灰度。
getRedComponent()	返回当前图形所使用的色彩中的红色成分。
getGreenComponent()	返回当前图形所使用的色彩中的绿色成分。
getBlueComponent()	返回当前图形所使用的色彩中的蓝色成分。
getFontMetrics()	返回当前图形所使用的字体规格。
getFont()	返回当前图形所使用的字体。
getLFont()	返回当前图形所使用的字体。(为兼容旧版)
setFont(int size)	设定当前图形所使用的字体大小。
setFont(String familyName, int style, int size)	设定当前图形所使用的字体名、样式以及大小。
setFont(LFont font)	设定当前图形所使用的字体。
setTransform(int transform, int width, int height)	以指定宽、高位条件，翻转画布为任意角度。
hitClip(int x, int y, int width, int height)	如果指定的矩形区域与当前的剪贴区域相交，则返回 true。
getClipBounds()	返回当前剪贴区域的边界矩形。
draw(Shape s)	使用当前 LGraphics 的设置勾画 Shape 的轮廓。可应用的呈现属性包括 Clip、Transform、Paint、Composite 和 Stroke 属性。
fill(Shape s)	使用 LGraphics 上下文的设置，填充 Shape 的内部区域。应用的呈现属性包括 Clip、Transform、Paint 和 Composite 属性。
fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)	用此图形上下文的当前颜色绘制圆角矩形的边框。矩形的左边和右边分别位于 x 和 x + width。矩形的顶边和底边位于 y 和 y + height。
fillTriangle(……)	此函数用以绘制单独或一组 Triangle 到屏幕当中，绘制时采取填充模式。
drawTriangle(……)	此函数用以绘制单独或一组 Triangle 到屏幕当中，绘制时仅勾画轮廓。
setXORMode(LColor color)	将此图形下文的绘图模式设置为在此图形上下文的当前颜色和新的指定颜色之间交替。这指定了逻辑像素操作以 XOR 模式执行，在此模式中像素在当前颜色和指定 XOR 颜色之间进行交替。
setPaintMode()	设置此图形上下文的绘图模式，以便通过此图形上下文中的当前颜色来改写目标。
setFill()	设置此图形的绘图模式为 FILL，即将 Android 的 Paint 设定为

	Style.FILL。
setStroke(Stroke s)	为 LGraphics 上下文设置 Stroke，这是个用于勾画针对 Shape 的 Stroke 对象。
rotate(double theta)	将当前的 LGraphics 进行旋转，并呈现出相对于前一原点旋转指定弧度的状态。事实上这近似于调用 transform。
rotate(double theta, double x, double y)	将当前的 LGraphics 进行旋转，呈现的变换是平移到指定位置旋转指定弧度，然后向回平移相同的距离。事实上这近似于调用 transform。
scale(double sx, double sy)	将当前的 LGraphics 进行缩放，并根据以前的比例按照指定的缩放系数来重新调整大小。
rectFill(int x, int y, int width, int height, LColor color)	以指定大小，指定颜色绘制方块（有填充）。
rectDraw(int x, int y, int width, int height, LColor color)	以指定大小，指定颜色绘制方块（无填充）。
rectOval(int x, int y, int width, int height, LColor color)	以指定大小，指定颜色绘制圆形（有填充）。
drawSixStart(LColor color, int x, int y, int r)	在指定位置绘制一颗六芒星，星大小根据 r 值决定。
drawTriangle(LColor color, int x, int y, int r)	在指定位置绘制一个正三角形，三角形大小根据 r 值决定。
drawRTriangle (LColor color, int x, int y, int r)	在指定位置绘制一个倒三角形，倒三角形大小根据 r 值决定。
drawBitmap(Bitmap bit, int x, int y)	将指定 Bitmap 绘制到指定位置。
drawBitmap(Bitmap bit, int x, int y, int anchor)	将指定 Bitmap 绘制到指定位置，并且按照 anchor 参数矫正显示的位置。（anchor 参数来自 LGraphics 中静态常量）
drawBitmap(Bitmap bit, int x, int y, int w, int h)	将指定 Bitmap 绘制到指定位置，并且扩充为指定大小。
drawBitmap(Bitmap bit, Matrix marMatrix, boolean filter)	将指定 Bitmap 以指定 Matrix 进行绘制，并且按照 filter 决定是否清晰化图像。
drawBitmap(Bitmap bit, Matrix marMatrix)	将指定 Bitmap 以指定 Matrix 进行绘制。
drawMirrorBitmap(Bitmap bit, int x, int y, boolean filter)	镜像翻转指定 Bitmap 到指定位置(有缓存) ，并且按照 filter 决定是否清晰化图像。
drawNotCacheMirrorBitmap(Bitmap bit, int x, int y,boolean filter)	镜像翻转指定 Bitmap 到指定位置(无缓存)，并且按照 filter 决定是否清晰化图像。
drawFlipBitmap(Bitmap bit, int x, int y, boolean filter)	水平翻转指定 Bitmap 到指定位置，并且按照 filter 决定是否清晰化图像。

drawReverseBitmap(Bitmap bit, int x, int y, boolean filter)	颠倒指定 Bitmap 到指定位置，并且按照 filter 决定是否清晰化图像。
drawBitmap(Bitmap bit, int x, int y, int w, int h, int x1,int y1, int w1, int h1)	将指定 Bitmap 绘制到当前可用的指定图像区域，动态地缩放图像使其符合目标绘制表面的指定区域。
drawBitmap(Bitmap bit, Rect r1, Rect r2)	将指定 Bitmap 绘制到当前可用的指定图像区域，动态地缩放图像使其符合目标绘制表面的指定区域。
drawImage(String fileName, int x, int y)	绘制指定路径图形资源到指定坐标。
drawImage(String fileName, int x, int y, int w, int h)	绘制指定路径图形资源到指定坐标为指定大小。
drawImage(LImage img, int x, int y)	将指定 LImage 绘制到指定位置
drawImage(LImage img, int x, int y, int anchor)	将指定 LImage 绘制到指定位置，并且按照 anchor 参数矫正显示的位置。（anchor 参数来自 LGraphics 中静态常量）
drawImage(LImage img, int x, int y, int w, int h)	将指定 LImage 绘制到指定位置，并且扩充为指定大小。
drawImage(LImage img, Matrix marMatrix, boolean filter)	将指定 LImage 以指定 Matrix 进行绘制，并且按照 filter 决定是否清晰化图像。
drawImage(LImage img, Matrix marMatrix)	将指定 LImage 以指定 Matrix 进行绘制。
drawMirrorImage(LImage img, int x, int y, boolean filter)	镜像翻转指定 LImage 到指定位置(有缓存) ，并且按照 filter 决定是否清晰化图像。
drawNotCacheMirrorBitmap(LImage img, int x, int y,boolean filter)	镜像翻转指定 LImage 到指定位置(无缓存)，并且按照 filter 决定是否清晰化图像。
drawFlipImage(LImage img, int x, int y, boolean filter)	水平翻转指定 LImage 到指定位置，并且按照 filter 决定是否清晰化图像。
drawReverseImage(LImage img, int x, int y, boolean filter)	颠倒指定 LImage 到指定位置，并且按照 filter 决定是否清晰化图像。
drawImage(LImage img, int x, int y, int w, int h, int x1,int y1, int w1, int h1)	将指定 LImage 绘制到当前可用的指定图像区域，动态地缩放图像使其符合目标绘制表面的指定区域。
drawRegion(Bitmap bit, int x_src, int y_src, int width,int height, int transform, int x_dst, int y_dst, int anchor)	从指定的 Bitmap 中截取指定位置图像后复制在指定区域，并且根据 transform 选择进行适当的旋转变换，按照 anchor 参数矫正显示的位置。（anchor 参数来自 LGraphics 中静态常量）
drawRegion(LImage src, int	从指定的 LImage 中截取指定位置图像后复制在指定区域，并且

x_src, int y_src, int width,int height, int transform, int x_dst, int y_dst, int anchor)	根据 transform 选择进行适当的旋转变换，按照 anchor 参数矫正显示的位置。（anchor 参数来自 LGraphics 中静态常量）
drawLine(int x1, int y1, int x2, int y2)	在此图形上下文的坐标系统中，使用当前颜色在点 (x1, y1) 和 (x2, y2) 之间画一条线。
drawRect(int x, int y, int width, int height)	绘制指定矩形的边框。
draw3DRect(Rectangle rect, LColor back, boolean down)	绘制指定矩形的 3D 突出显示边框。矩形的边是突出显示的，从左上角看上去呈斜面并加亮。
draw3DRect(int x, int y, int width, int height, boolean raised)	绘制指定矩形的 3D 突出显示边框。矩形的边是突出显示的，从左上角看上去呈斜面并加亮。
drawOval(int x, int y, int width, int height)	绘制圆形的边框。得到一个圆或椭圆，它的位置恰好适合放在由 x、y、width 和 height 参数指定的矩形内。
drawPolygon(int[] xpoints, int[] ypoints, int npoints)	绘制一个由 x 和 y 坐标数组定义的闭合多边形。每对 (x, y) 坐标定义了一个点。
drawPolygon(Polygon p)	绘制由指定的 Polygon 对象定义的多边形边框。
drawPolyline(int[] xpoints, int[] ypoints, int npoints)	绘制由 x 和 y 坐标数组定义的一系列连接线。
drawRoundRect(int x, int y, int width, int height,int arcWidth, int arcHeight)	用此图形上下文中所指定的颜色绘制圆角矩形的边框。
drawArc(int x, int y, int width, int height, int sa, int ea)	绘制一个覆盖指定矩形的圆弧或椭圆弧边框。
drawBytes(byte[] message, int offset, int length, int x, int y)	使用此图形上下文的字体和颜色绘制由指定的 byte 数组给定的文本。首字符的基线位于此图形上下文坐标系统的 (x, y) 位置处。
drawChars(char[] message, int offset, int length, int x, int y)	使用此图形上下文的字体和颜色绘制由指定 char 数组给定的文本。首字符的基线位于此图形上下文坐标系统的 (x, y) 位置处。
drawChar(char message, int x, int y)	使用此图形上下文的字体和颜色绘制字符中给定的文本。
drawString(String message, int x, int y)	使用此图形上下文的字体和颜色绘制字符串中给定的文本。
drawString(String message, int x, int y, int anchor)	使用此图形上下文的字体和颜色绘制字符中给定的文本，并按照 anchor 参数矫正显示的位置。（anchor 参数来自 LGraphics 中静态常量）
drawSubstring(String message, int offset, int len, int x,int y, int anchor)	使用此图形上下文的字体和颜色绘制字符中指定区域间给定的文本，并按照 anchor 参数矫正显示的位置。（anchor 参数来自 LGraphics 中静态常量）
drawSubString(String	使用此图形上下文的字体和颜色绘制字符中指定位置间给定的

message, int x, int y, int w, int h,int anchor)	文本，并按照 anchor 参数矫正显示的位置。（anchor 参数来自 LGraphics 中静态常量）
draw3DString(String message, int x, int y, LColor c)	使用此图形上下文的字体和 LColor 提供的颜色以伪 3D 模式绘制字符中给定的文本。
drawCenterString(String message, int x, int y)	使用此图形上下文的字体和颜色居中绘制指定文字。
drawShadeString(String message, int x, int y, LColor color,LColor color1, int k)	使用此图形上下文的字体和 LColor 提供的颜色以 k 指定的边框大小绘制指定带阴影文字。
drawCenterShadeString(String message, int x, int y,LColor color, LColor color1, int k)	使用此图形上下文的字体和 LColor 提供的颜色以 k 指定的边框大小居中绘制指定带阴影文字。
drawCenterRoundedString(String message, int x, int y,LColor color, LColor color1)	使用此图形上下文的字体和 LColor 提供的颜色居中绘制指定带边框文字。
drawStyleString(String message, int x, int y, int color,,int color1)	使用此图形上下文的字体和 int 参数所提供的像素颜色提供的颜色绘制指定带边框文字。
drawStyleString(String message, int x, int y, LColor c1,LColor c2)	使用此图形上下文的字体和 LColor 提供的颜色绘制指定带边框文字。
drawRGB(int[] colors, int offset, int stride, int x, int y,int width, int height, boolean hasAlpha)	该方法用于在规定区域呈现一系列独立于设备的 RGB 和透明的值 rgbData,rgbData 是一个 ARGB 值数组,offset 是数组中第一个 ARGB 值的索引,stride 是在 rgbData 数组连续行相应的像素间的相对的 offset,x 是显示区域的水平起点,y 是显示区域的垂直起点,width 是显示区域的宽,height 是显示区域的高。而 hasAlpha 的设定依据在于,如果 rgbData 有透明通道的话,则为 true;如果所有像素都是不透明的,则为 false。
drawClear()	清空屏幕画布。（变为纯黑色）
setBackground(LColor color)	设置 LGraphics 上下文的背景色,背景色用于清除所有图形区域。
getBackground()	返回 LGraphics 背景色。
fillArc(int x, int y, int width, int height, int sa, int ea)	填充覆盖指定矩形的圆弧或椭圆弧。
fillOval(int x, int y, int width, int height)	使用当前颜色填充外接指定矩形框的椭圆。
fillPolygon(int[] xpoints, int[] ypoints, int npoints)	填充由 x 和 y 坐标数组决定的闭合多边形。
fillPolygon(Polygon p)	用图形上下文的当前颜色填充由指定的 Polygon 对象决定的多边形。
fillRect(int x, int y, int	填充指定的矩形。

width, int height)	
fillAlphaRect(int x, int y, int w, int h, LColor c)	以指定颜色填充指定矩形区域。
fillAlphaRect(int x, int y, int w, int h, int pixel)	以指定像素填充指定矩形区域。
fillTriangle(int x1, int y1, int x2, int y2, int x3, int y3)	使用此图形上下文的颜色填充一个指定区域内的三角形。
fill3DRect(int x, int y, int width, int height, boolean raised)	绘制一个用当前图形上下文颜色填充的 3D 突出显示矩形。
fillRoundRect(int x, int y, int width, int height,int arcWidth, int arcHeight)	用当前颜色填充指定的圆角矩形。
draw3DRect(Rectangle rect, LColor back, boolean down)	绘制一个用指定颜色填充的 3D 突出显示矩形。
draw3DRect(int x, int y, int width, int height, boolean raised)	绘制一个用当前图形上下文颜色填充的 3D 突出显示边框。
clip(Shape s)	使用当前 LGraphics 的设置截取出 Shape 的轮廓。可应用的呈现属性包括 Clip、Transform、Paint、Composite 和 Stroke 属性。
draw(Path s)	使用当前 LGraphics 的设置勾画 Path 的轮廓。可应用的呈现属性包括 Clip、Transform、Paint、Composite 和 Stroke 属性。
getClipWidth()	获得当前剪辑区的宽度。
getClipHeight()	获得当前剪辑区的高度。
getClipX()	获得当前剪辑区的 x 坐标。
getClipY()	获得当前剪辑区的 y 坐标。
setTransform(AffineTransform aff)	重写 LGraphics 上下文中的 Transform，将画面旋转为指定样式。
getTransform()	返回当前 LGraphics 的 Transform 设置。
clearRect(int x, int y, int width, int height)	通过使用当前绘图表面的背景色进行填充来清除指定的矩形。
clearScreen(int x, int y, int width, int height)	通过使用当前绘图表面的背景色进行填充来清除指定的矩形。（兼容旧版方法）
copyArea(int x_src, int y_src, int width, int height,int x_dest, int y_dest, int anchor)	将组件的区域复制到由 x_dest 和 y_dest 指定的距离处，并按照 anchor 参数矫正显示的位置。（anchor 参数来自 LGraphics 中静态常量）
copyArea(int x, int y, int width, int height, int dx, int dy)	将组件的区域复制到由 dx 和 dy 指定的距离处。
clipRect(int x, int y, int width, int height)	将当前剪贴区与指定的矩形相交，得到的剪贴区域是当前剪贴区域和指定矩形的交集。
shear(double shx, double	将当前 LGraphics 的 Transform 与剪裁区域转换后连接，呈现出相

shy)	对于以前的位置所指定的乘式剪裁。
setStrokeStyle(int style)	在进行画线、弧形、矩形和圆角矩形时，设定绘制模式，该方法对文本和图像的绘制无效。该函数使用的绘制模式只有两种，SOLID 和 DOTTD。（相关参数来自 LGraphics 中静态常量）
getStrokeStyle()	获得针对画线、弧形、矩形和圆角矩形时所设定的绘制模式
transform(AffineTransform aff)	使用此 LGraphics 中的 Transform 组合 AffineTransform 对象进行图形翻转。
translate(int x, int y)	该函数用于重新设定图形坐标系统原点，以后的操作也都针对此原点进行。
setClip(int x, int y, int width, int height)	将当前的剪贴区设置为给定坐标指定的矩形。
setClip(Shape shape)	将当前的剪贴区设置为给定 Shape 指定的形状。
setClip(Rect rect)	将当前的剪贴区设置为给定 Rect 指定的形状。
getClip()	返回当前图形的剪切区。
setBitmap(Bitmap bit)	设定当前图形使用的内部 Bitmap。
setPaint(Paint paint)	设定当前图形使用的内部 Paint。
getCanvas()	获得当前图形使用的内部 Canvas。
getBitmap()	获得当前图形使用的内部 Bitmap。
getPaint()	获得当前图形使用的内部 Paint。
dispose()	关闭当前 LGraphics 并释放所有资源。
toString()	返回当前 LGraphics 实际类名。
isClose()	判断当前图形资源是否已被关闭。

11、 org.loon.framework.android.game.core.graphics.filter

在此包下共有类 4 个，分别为 ImageFilter、ImageFilterExecute、ImageFilterFactory、ImageFilterType，他们的作用在于过滤 Bitmap 以及 LImage 图形，令指定的图形呈现出例如黑白化、去色化以及纯色化等效果，其中核心执行类为 ImageFilterFactory，而默认效果的静态参数存放于 ImageFilterType 当中。由于此部分并非核心功能，暂无详述，目前请以源码为主。

12、 org.loon.framework.android.game. core.graphics.geom

在此包下共有类 40 个，提供了大量与二维几何形状相关的对象定义以及执行操作类，可以通过此包与 LGraphics 配合，生成出我们所需的任意复杂图形（诸如饼状图、股票走势图等）。此外，由于该包完全移植自 openJDK 的 java.awt.geom 包（兼顾 LGame-J2SE 版使用习惯），可以直接参阅相关 Java 文档，暂无详述，目前请以 Java 相关文档及源码为主。

13、 org.loon.framework.android.game.core.graphics.window

在此包下共有类 10 个，他们分别为 LButton、LMessage、LPanel、LPaper、LPicture、LSelect、LUI、UIConfig、UIFactory、UIStatic，其中前六项为可以直接使用的 LGame 组件。值得一提的是，由于 LGame-Android 版中除 LButton 外所有组件都直接继承自 LContainer，因此它们可以直接 add 其它组件到其中混合使用。比如在 LPaper 上添加 LButton 制作出可拖动的选择框，在 LMessage 上添加 LPaper 用以显示某个角色头像或动画，都是可以轻易实现的。

此外，由于 LGame 引擎采取纯绘制完成，所以组件位置可以叠加，可以相互覆盖，可以设定为指定的透明度，可以自由拖拽。

（1） LButton

LButton 是 LGame 引擎提供的原生组件之一（纯绘制），也是唯一不允许直接触摸屏拖拽的 LGame-Android 组件（因为是按钮），他的作用是将您选择的任意图片分解为按钮并显示于手机窗体之上。当然，他可以自动获得并响应触摸屏事件。

函数名	函数说明
setImages(LImage[] images)	插入一组图像（1-4 张皆可，超出忽略不计），并将它们作为 LButton 的按纽显示用图，以分别对应按钮的未点击、点击、触摸屏划过以及其它状态。
update(long timer)	变更按钮时间点状态。（组件管理器会自动触发此事件，如果想在管理器外使用请自行设置触发条件）
isTouchPressed()	对当前按钮按下触摸屏。
isTouchOver()	对当前按钮结束触摸。
setText(String st)	设置当前按钮的显示文本。
getText()	返回当前按钮的显示文本。
doClick()	当于触摸屏之上点击或释放当前按钮时，将触发此事件，可以通过重载此函数完成相应操作。
downClick()	仅在点击触摸屏时有效，可以通过重载此函数完成相应操作。
upClick()	仅在释放触摸屏时有效，可以通过重载此函数完成相应操作。
isException()	获得当前按钮事件是否发生了异常。
setFont(LFont font)	设定当前按钮所使用的字体。
getFont()	返回当前按钮所使用的字体。
setFontColor(LColor fontColor)	设定当前按钮所使用的字体颜色。
getFontColor()	返回当前按钮所使用的字体颜色。
setOffsetLeft(int offsetLeft)	设定当前按钮自按钮左侧起的文本偏移值。（默认为 0）
getOffsetLeft()	返回当前按钮自按钮左侧起的文本偏移值。
setOffsetTop(int offsetTop)	设定当前按钮自按钮顶端起的文本偏移值。（默认为 0）
getOffsetTop()	返回当前按钮自按钮顶端起的文本偏移值。
getUIName()	返回当前组件名称。

（2） LMessage

LMessage 是 LGame 引擎提供的原生组件之一（纯绘制），被用来以逐字打印的方式显示我们需要的文本信息给用户获悉，并且允许 add 任意组件到此组件当中。此外，如果我们将它的 setLocked 方法设定为 False（默认为 True），则可以利用触摸屏随意拖拽该组件到屏幕随意位置。

函数名	函数说明
complete()	强制完成逐字打印效果。在我们不需要 LMessage 逐字打印指定信息时，可以调用此函数直接显示全部数据给用户。

setLeftOffset(int left)	设定自消息框左侧起的文本偏移值。（默认为 0）
getLeftOffset()	返回自消息框左侧起的文本偏移值。
setTopOffset(int top)	设定自消息框顶端起的文本偏移值。（默认为 0）
getTopOffset()	返回自消息框顶端起的文本偏移值。
setMessageLength(int messageLength)	设定消息框每行所允许的字符数量，当超过此数量时，文本打印将自动换行。（在文本中穿插\n 符号可提前换行）
getMessageLength()	返回消息框每行所允许的字符数量。
setTipIcon(String fileName)	设定指定路径对应的图片资源为文字悬停时的后缀显示图。
setTipIcon(LImage icon)	设定指定图片资源为文字悬停时的后缀显示图。
setNotTipIcon()	清空图文字悬停时的后缀显示图。
setDelay(long delay)	设定逐字打印时的文字间隔时间。
isComplete()	判断当前消息框的所有内容是否已经逐字打印完毕。
setMessage(String context)	设定消息框的显示内容。
setMessage(String context, boolean isComplete)	设定消息框的显示内容，当 isComplete 为 True 时，将取消逐字打印效果。
doClick()	当于触摸屏之上点击或释放当前消息框时，将触发此事件，可以通过重载此函数完成相应操作。
update(long elapsedTime)	变更当前消息框时间点。
setPauseIconAnimation(Animation animation)	设定当前消息框打印完毕时的停止时动画。（默认为无，可以用来显示一些诸如翻书之类的小动画）
setPauseIconAnimationLocation(int dx, int dy)	设定当前消息框打印完毕时的停止时动画显示位置，该位置不能超出消息框所在。
setFontColor(LColor fontColor)	设定当前消息框文字颜色。（也可以通过“<r 文字/>”方式输入字符串以改变单个文字颜色）
getFontColor()	返回当前文字颜色。
setMessageFont(LFont messageFont)	设定当前消息框文字字体。
getMessageFont()	返回当前消息框文字字体。
setLocked(boolean locked)	设定是否锁定当前消息框，默认为 True，当为 False 时消息框可以自由拖拽。
isLocked()	判断是否锁定了当前消息框。
getUIName()	返回当前组件名称。

（3） LPaper

LPaper 就组件而言是一张白纸，本身并不显示任何数据，也正因如此，无论什么数据我们都可以置于其中进行显示。此外，如果我们将它的 setLocked 方法设定为 False（默认为 True），则可以利用触摸屏随意拖拽该组件到屏幕随意位置。

函数名	函数说明
setAnimation(Animation animation)	插入指定的动画到当前白纸组件中。
getAnimation()	返回当前白纸组件中的动画类。

addAnimationFrame(SpriteImage image, long timer)	插入一个指定精灵图片到白纸组件动画显示中，并设置其对应帧播放速度。
addAnimationFrame(String fileName, long timer)	插入指定路径的图片资源到白纸组件动画显示中，并设置其对应帧播放速度。
addAnimationFrame(LImage image, long timer)	插入指定图像到白纸组件动画显示中，并设置其对应帧播放速度。
doClick()	当于触摸屏之上点击或释放当前白纸组件时，将触发此事件，可以通过重载此函数完成相应操作。
update(long elapsedTime)	变更当前白纸组件时间点。
setLocked(boolean locked)	设置是否锁定当前白纸组件。
isLocked()	判断是否锁定了当前白纸组件。
getUIName()	返回当前组件名。

（4） LPicture

LPicture 是一个被用来显示指定图片的 LGame 组件，他可以将 APK 中或网络上的图像资源显示在手机之上。此外，如果我们将它的 setLocked 方法设定为 False（默认为 True），则可以利用触摸屏随意拖拽该组件到屏幕随意位置。

函数名	函数说明
notBackground()	让图片框不显示任何图片。
loadImage(final String fileName, final boolean transparency)	载入 APK 中的指定图像到图片框中，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（有缓存）
loadWebImage(String url, boolean transparency)	载入指定路径的网络图片资源到图片框中，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（有缓存）
loadImage(final LImage img)	载入指定图片到图片框中。
getImage()	返回当前图片框中图像。
doClick()	当于触摸屏之上点击或释放当前图片框时，将触发此事件，可以通过重载此函数完成相应操作。
update(long elapsedTime)	变更图片框时间点。
getStates()	返回图片框中网络资源的加载状态。
setLocked(boolean locked)	设定是否锁定图片框。
isLocked()	判定是否锁定了图片框。
getUIName()	返回当前组件名称。

（5） LSelect

LPicture 是一个被用来显现选择框的 LGame 组件，他可以用来获得及记录用户于触摸屏上的拖拽选择，并反馈给 LGame 引擎。此外，如果我们将它的 setLocked 方法设定为 False（默认为 True），则可以利用触摸屏随意拖拽该组件到屏幕随意位置。

函数名	函数说明
setLeftOffset(int left)	设定自选择框左侧起的文本偏移值。（默认为 0）
getLeftOffset()	返回自选择框左侧起的文本偏移值。
setTopOffset(int top)	设定自选择框顶端起的文本偏移值。（默认为 0）
getTopOffset()	返回自选择框顶端起的文本偏移值。
getResult()	返回当前选择结果的字符串形式。
getResultIndex()	返回当前的选择结果的整型数值。（索引由 0 开始）
setDelay(long timer)	设定当前选择框时间点刷新速度。
getDelay()	返回当前选择框时间点刷新速度。
setMessage(String message, List list)	设定选择框的标题及显示数据。
setMessage(String[] selects)	设定选择框的显示数据。
setMessage(String message, String[] selects)	设定选择框的标题及显示数据。
update(long elapsedTime)	变更选择框时间点。
doClick()	当于触摸屏之上点击或释放当前选择框时，将触发此事件，可以通过重载此函数完成相应操作。
isClick()	获得当前选择框是否遭到点击。
setFontColor(LColor fontColor)	设定选择框的字体颜色。
getFontColor()	返回选择框的字体颜色。
setMessageFont(LFont messageFont)	设定选择框的字体样式。
getMessageFont()	返回选择框的字体样式。
setLocked(boolean locked)	设定是否锁定当前选择框。
isLocked()	判定是否锁定了当前选择框。
setCursor(LImage cursor)	设定当前选择框所使用的游标。
setCursor(String fileName)	从指定路径载入图片作为选择框的游标。
setNotCursor()	取消选择框的游标。
getCursor()	返回当前选择框的游标。
setBuoyage(LImage buoyage)	设定当前选择框的文字浮标。
setBuoyage(String fileName)	从指定路径载入图片作为选择框的文字浮标。
setFlashBuoyage(boolean flashBuoyage)	设定是否启用文字浮标。
isFlashBuoyage()	判定是否启用了文字浮标。
getUIName()	返回当前组件名。

14、 org.loon.framework.android.game.

core.graphics.window.achieve

在此包中共有类 3 个（相较 LGame-J2SE 减少了很多），他们分别为 IButton、IPanel 以及 IPrint，此部分内容为 LGame 标准组件的配置包。由于此部分通常并不直接提供给用户使用，故暂无详述，目前请以源码为主。

15、 org.loon.framework.android.game.core.resource

在此包中共有类 6 个，他们分别为 LPKHeader、LPKResource、LPKTable、LPKType、Resources、ZipResource，其中最为核心的是 Resources 类以及 LPKResource 类，它们是 LGame 引擎最常使用的资源加载器。

(1) Resources

函数名	函数说明
getClassLoader()	返回一个针对当前 APK 环境的 ClassLoader 管理器。
getClassLoader(Class clazz)	返回一个根据指定类获得的 ClassLoader 管理器。
getNames()	返回一组 Iterator 迭代器，其中包含有所有缓存资源的名称。
contains(String resName)	检查指定资源是否在缓存中存在。
remove(String resName)	删除指定的缓存资源。
destroy()	注销所有的缓存资源。
openResource(String resName, ClassLoader c)	通过指定的 ClassLoader 管理器，加载 APK 资源到输入流中。
openResource(String resName)	从当前 APK 中获得指定路径资源的输入流。
getResource(String resName)	从当前 APK 中获得指定路径资源，并且转化为 ArrayByte 返回（ArrayByte 是一个内部封装了比特数组的集合类），此方法存在缓存。
getNotCacheResource(String resName)	从当前 APK 中获得指定路径资源，并且转化为 ArrayByte 返回（ArrayByte 是一个内部封装了比特数组的集合类），此方法无缓存。
getResourceAsStream(String fileName)	加载指定路径资源为输入流。（有缓存）
getNotCacheResourceAsStream(String fileName)	加载指定路径资源为输入流。（无缓存）
getDataSource(InputStream is)	将指定输入流转化为比特数组并返回。（无缓存）
getResource(Class clazz, String resName)	获得指定类所在地对应的指定资源。（无缓存）
getHttpStream (String uri)	从网络获得指定资源并转化为比特数组后返回。(无缓存)

（2） LPKResource

这是一个读取 LGame 引擎压缩包的工具类（“LPK”格式），他的作用是从一整个压缩文件中分解出需要的资源。此外，LGame-Android 版无法直接压缩 LPK 文件（因为是手机环境），对应的压缩工具类可在 LGame-J2SE 版 Utils 工具包中获得。

函数名	函数说明
openResource(String fileName, String resName)	从 APK 中读取指定的 LPK 文件，并从中分离出需要的数据为比特数组。
openImage(String fileName, String resName)	从 APK 中读取指定的 LPK 文件，并从中分离出需要的数据为 LImage 格式。
getLPKInfo(String resName)	获得指定 LPK 资源的配置信息。
readHeader(DataInputStream dis)	读取指定 LPK 资源头文件。
readLPKTable(DataInputStream dis, int fileTableNumber)	分解指定输入流中的 LPK 资源为 LPK 表格信息数组。
readFileFromPak(DataInputStream dis,LPKHeader header,LPKTable fileTable)	分解指定输入流以及指定 LPK 头文件和指定 LPK 表格的 LPK 资源为比特数组。

16、 org.loon.framework.android.game.core.store

在此包中共有类 12 个，他们分别为 InvalidRecordIDException、RecordComparator、RecordEnumeration、RecordFilter、RecordListener、RecordStore、RecordStoreException、RecordStoreFullException、RecordStoreNotFoundException、RecordStoreNotOpenException、RecordStoreSqlLite、RecordStoreSqlLiteEnumeration，他们严格按照 MicroEmulator 所提供的接口，仿照 J2ME 中相关功能实现（Android 版的具体实现方式为内部封装 SQLite），可以用于存储 LGame 引擎中游戏数据或其它需要存储的信息。由于此部分内容较多，而且可以参考相关 J2ME 相关文档，所以暂无详述，具体信息目前可以通过网路中的 J2ME 文档查找同名类获得。此外，如果您没有 J2ME 经验，又或者想更加简便的使用此部分功能，可以直接使用 utils 包中的 RecordStoreUtils 类。

17、 org.loon.framework.android.game.core.timer

在此包中共有类 4 个，他们分别为 LTimer、LTimerContext、NanoTimer、SystemTimer，是 LGame 引擎提供的游戏时间点记时工具类，不过需要注意的是，以上这些类的作用只是“记录时间经过”，而并没有自动获得时间变化的能力，所以他们通常会与线程一起使用。其中最为常用的是 LTimer。

（1） LTimer

函数名	函数说明
action(long elapsedTime)	此函数用以记录并统计时间经过，当积累的经过时间与初始化所设定的 LTimer 时间完全一致或超过时，此方法将返回 True 并重新开始计算时间，否则将始终返回 False。
refresh()	释放已经统计的时间，将已经过时间归零。
setEquals(LTimer other)	设定当前计时器时间设定等于指定计时器的时间设定。
setActive(boolean bool)	设定当前计时器是否正在活动。
isActive()	判断当前计时器是否正在活动。
start()	激活当前计时器。
stop()	停止当前计时器。

setDelay(long delay)	设置每步计时间延迟。
getDelay()	获得每步计时间延迟。
setCurrentTick(long tick)	强行设置当前已经过时间。
getCurrentTick()	获得当前计时器已经过时间。

18、 org.loon.framework.android.game.media.sound

在此包中共有类两个，分别为 AssetsSound 以及 AssetsSoundManager，他们的作用在于读取并播放管理 APK 中的音频文件。由于 Android 对于音频文件的特殊限制，我们无法从任意目录读取音频，而只能通过 Assets 文件夹获得，所以 LGame 引擎将他们做了如此命名，以尽量防止用户的错误操作。其中最常用的类是 AssetsSound。

（1） AssetsSound

函数名	函数说明
play()	播放音频文件。
setDataSource(String file)	设定 APK 中的指定音频文件到 AssetsSound 中。（当然，该文件必须在 Assets 之下）
play(int vol)	以指定音量播放音频文件
loop()	循环播放音频文件。
stop()	停止音频文件播放。
reset()	刷新（重置）音频播放器。
release()	释放音频播放器。
setVolume(int vol)	设定播放器音量。
stopPlayer()	强制停止音频播放器播放。（硬性关闭内部的 MediaPlayer）
getName()	获得当前播放器名称。

19、 org.loon.framework.android.game.utils

在此包下共有类 10 个，他们分别是 CollectionUtils、CollisionUtils、CompressionUtils、FileUtils、GraphicsUtils、MultitouchUtils、NumberUtils、PassWordUtils、RecordStoreUtils、StringUtils，他们为 LGame 引擎提供了大量的辅助操作功能。在他们当中，用户最为常用的应该是 GraphicsUtils 类，这是一个针对 Android 环境的图形资源管理类。

(1) GraphicsUtils

函数名	函数说明
loadBitmap(InputStream in, boolean transparency)	从指定输入流载入一个 Bitmap，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置(Options 参数)，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色

	彩模式是主要决定因素。
loadBitmap(String resName, boolean transparency)	从指定路径载入一个 Bitmap，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。
loadScaleBitmap(String resName, int width,int height)	从指定路径载入 Bitmap，并依据指定大小返回给用户。
loadScaleImage(String resName, int width,int height)	从指定路径载入 LImage，并依据指定大小返回给用户。
loadImage(InputStream in, boolean transparency)	从指定输入流载入一个 LImage，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。
load8888Image(InputStream in)	从指定输入流以 ARGB8888 格式载入 LImage。
load8888Image(String fileName)	从指定路径以 ARGB8888 格式载入 LImage。
loadImage(byte[] buffer, boolean transparency)	从指定比特数组载入一个 LImage，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。
loadImage(byte[] imageData, int imageOffset,int imageLength, boolean transparency)	从指定比特数组的指定数据范围内载入一个 LImage，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。
loadImage(String innerFileName, boolean transparency)	从指定路径载入一个 LImage，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（有缓存）
loadImage(String innerFileName)	从指定路径载入一个 LImage。（有缓存）
LImage loadNotCacheImage(String innerFileName,boolean	从指定路径载入一个 LImage，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选

transparency)	配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（无缓存）
loadNotCacheImage(String innerFileName)	从指定路径载入一个 LImage。（无缓存）
loadWebImage(String string, boolean transparency)	从网络路径载入一个 LImage，并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（无缓存）
loadSequenceImages(String fileName, String range,boolean transparency)	从指定路径获得一组获得序号连续的图片到 LImage 数组，range 指定图片范围，如"1-2"。并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（有缓存）
getFlipHorizintalImage2D(LImage[][] pixels)	水平翻转一组 LImage 图像。
rotateImage(LImage image)	翻转一张 LImage 图像（180 度）
rotateImage(LImage image, int angdeg,boolean d)	翻转一张 LImage 图像为指定角度，当 d 为 True 时逆像翻转，为 False 时正向翻转。
createShapeImage(Shape shape, LColor c1, LColor c2)	依旧 Shape 创建一个指定形状和填充色的 LImage
drawClipImage(LImage image, int objectWidth,int objectHeight, int x1, int y1, int x2, int y2, Config config)	以指定参数，指定格式剪切指定 LImage 图像。
drawClipImage(final LImage image, int objectWidth,int objectHeight, int x1, int y1, int x2, int y2)	以指定参数剪切指定 LImage 图像。
drawClipImage(final LImage image, int objectWidth,int objectHeight, int x1, int y1, int x2, int y2, boolean flag)	以指定参数剪切指定 LImage 图像,flag 为 True 时剪切为 ARGB4444 格式，为 False 时剪切为 RGB565 格式。
drawClipImage(final LImage image, int objectWidth,int objectHeight, int x, int y, boolean flag)	以指定参数剪切指定 LImage 图像，flag 为 True 时剪切为 ARGB4444 格式，为 False 时剪切为 RGB565 格式。
drawClipImage(final LImage image, int objectWidth,int objectHeight, int x, int y,	以指定参数，指定格式剪切指定 LImage 图像。

Config config)	
drawClipImage(final LImage image, int objectWidth,int objectHeight, int x, int y)	以指定参数剪切指定 LImage 图像。
drawCropImage(final LImage image, int x, int y,int objectWidth, int objectHeight)	以指定参数剪切指定 LImage 图像（这个的 x、y 参数在前，与其它的差异仅在使用习惯问题）。
getSplit2Images(String fileName, int row, int col,boolean isFiltrate, boolean transparency)	以指定参数分解指定路径的指定图像。并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（有缓存）
getSplit2Images(String fileName, int row, int col,boolean transparency)	以指定参数分解指定路径的指定图像。并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（有缓存）
getSplit2Images(LImage image, int row, int col,boolean isFiltrate)	以指定参数分解指定 LImage 图像。
getResize(LImage image, int w, int h)	缩放指定 LImage 图像。
getResize(Bitmap image, int w, int h, boolean flag)	缩放指定 LImage 图像，,flag 为 True 时缩放为 ARGB4444 格式，为 False 时缩放为 RGB565 格式。
getResize(Bitmap image, int w, int h)	缩放指定 Bitmap 图像。
resizeBitmap(Bitmap bmp, int w, int h)	缩放指定 Bitmap 图像。与同类函数差异在于内部缩放使用 BitmapDrawable 进行。
getMatrix()	返回一个静态的 Matrix。
getSplitImages(String fileName, int row, int col,boolean transparency)	切分指定路径图像。并根据 transparency 尝试以 RGB565（为 False 时）或 ARGB4444（为 True 时）显示。当然，由于 transparency 传递给 Android 后，只是修改了加载时的首选配置，而没有强制改变文件格式，所以图片并非一定会以 RGB565 或 ARGB4444 格式进行显示，图片本身的色彩模式是主要决定因素。（有缓存）
getSplitImages(LImage image, int row, int col)	切分指定路径图像。（有缓存）
copy(LImage target, LImage source)	拷贝指定图像到目标图形中。
rotate(Bitmap bmp, float degrees)	旋转指定 Bitmap 图像为指定角度。
rotate(LImage img, float	旋转指定 LImage 图像为指定角度。

degrees)	
reflect(Bitmap originalImage)	为指定 Bitmap 图像添加反射光效果。
fitBitmap(Bitmap baseImage, int width, int height)	矫正指定 Bitmap 为成比例大小后返回新的图片。
fitImage(LImage image, int width, int height)	矫正指定 LImage 为成比例大小后返回新的图片。
calculateFitBitmap(Bitmap baseImage, int width,int height, Point receiver)	以指定参数矫正指定的 Bitmap 大小。
loadAsPNG(String recordStore, String resourceName)	从 RecordStore 中读取一张指定位置的 PNG 格式图形文件。
saveAsPNG(String recordStore, String resourceName,LImage image)	保存一张 LImage 到 RecordStore 当中的指定位置去。
saveAsPNG(Bitmap bitmap, String fileName)	保存一张 Bitmap 到 RecordStore 当中去。
saveAsPNG(LImage image, String fileName)	保存一张 LImage 到 RecordStore 当中去。
hashBitmap(Bitmap bitmap)	获得指定 Bitmap 的 hash 数值（此为快速截取方式，近似图有可能出现重复）。
destroy()	注销所有的缓存资源。

20、 org.loon.framework.android.game.utils.collection

在此包中仅有 2 个集合类，他们分别是 ArrayByte 和 ArrayMap，用以通过顺序方式管理对应集合。由于此部分并非核心类，暂时不提供文档说明，目前请以实际代码为主。

21、 org.loon.framework.android.game.physics

在此包中含有类 18 个，分别是 CollisionEvent、LBody、LShape、LWorld、LWorldListener、MathUtils、PhysicsListener、PhysicsObject、PhysicsPolygon、PhysicsScreen、PhysicsUtils、PolygonBasedShape、PolygonDef、PolygonSprite、PolygonType、PrimitiveShape、Rectangle、WorldBox，他们被用以管理 Box2D 的 JNI 封装（Box2D 的 JNI 实现部分来自 libgdx 项目），其中最重要的是 PhysicsScreen 以及 PhysicsObject。

（1） PhysicsScreen

PhysicsScreen 继承自 Screen，其本身可以认为是一个自动刷新画面结果的绘图器，内部封装有针对 Box2D 的 World，世界大小默认即屏幕大小（可调），并提供了绑定任何事物到 PhysicsObject 的能力。其中涉及 Box2D 的部分（虽然 libgdx 的 JNI 封装并不完整，但大体使用方式是一致的），请自行参阅 Box2D 文档，暂不赘述。另外，在调试物理游戏时，个人建议使用真机或者运行环境硬件性能足够高的虚拟机。

函数名	函数说明
begineContactListener()	启动碰撞监听，当调用此函数后，物理精灵便可以获得自身的碰撞结果并进行反馈。
endContactListener()	关闭碰撞监听，当调用此函数后，物理精灵将不再接收碰撞信息，LGame 中默认此项为关闭状态。
find(int x, int y)	查找指定位置的物理精灵，如果存在返回一个 PhysicsObject，不存在返回 NULL。
find(int x, int y, Object tag)	查找指定位置，并具有指定 tag 的物理精灵，如果存在返回一个 PhysicsObject，不存在返回 NULL。
bindTo(……)	这是一系列函数集合，内部封装有所有 PhysicsObject 的构造函数，用以通过指定形式绑定形状、图片或 Body 到 PhysicsObject 当中。
addObject(PhysicsObject o)	添加一个物理精灵到 PhysicsScreen 当中。
removeObject(PhysicsObject o)	函数一个 PhysicsScreen 当中的物理精灵。
onDown(MotionEvent e)	当触发屏幕点击事件时，此函数会被调用。
onMove(MotionEvent e)	当触发屏幕移动事件时，此函数会被调用。
onUp(MotionEvent e)	当触发屏幕放开事件时，此函数会被调用。
createBody(BodyDef bodyDef)	创建一个以指定 BodyDef 描述的 Body。
destroyBody(Body body)	注销一个指定的 Body。
createJoint(JointDef def)	创建一个以指定 JointDef 描述的 Joint。
destroyJoint(Joint joint)	注销一个指定的 Joint。
getWorld()	返回当前正在使用的物理世界。
setGravity(int x, int y)	设定当前世界的重力。（即图形移动方向与每次移动多少像素）
getGravity()	返回当前世界的重力。
paint(LGraphics g)	绘制任意图形到当前 Screen 之上。
update(LTimerContext t)	每次屏幕自动刷新后执行，获得相关的时间参数。
onCollisionEvent(CollisionEvent e)	此函数用以获得物理精灵碰撞事件，当开启碰撞监听后可以重载使用此函数。
setWorldBox(……)	设定当前物理世界的大小。
setPhysicsFPS(long physicsFPS)	设定当前物理世界的刷新速度，针对不同机型，此参数有微调的必要。

（1） PhysicsObject

PhysicsObject 继承自 ISprite，它本身也是一个精灵，但地位却比较特殊，因为它的主要作用在于同 PhysicsScreen 联动，所以其提供的大多数构造方法不能在 PhysicsScreen 以外的 Screen 中使用。PhysicsObject 另一个比较特殊的地方在于，它能够直接将图像转化为形状（我诅咒 Box2D 万恶的凸多边形限制）并参与碰撞，而无需手动设定。此外，BodyType 与 Body 提供的参数皆可在 PhysicsObject 当中进行设置（此部分函数太多不一一细数，具体请参见源码），但只有调用 make 后才会生效。

函数名	函数说明
make()	在 PhysicsObject 中这是一个非常重要的函数，只有执行此函数后，物理精灵才会根据设置产生出可用的 Body。
setType(BodyType type)	设定当前物理精灵（即内部 Body）的 BodyType 类型。
update(long elapsedTime)	重载此函数，可以获得当前画面距离上次刷新的时间。
setUserData(Object object)	设定当前物理精灵（即内部 Body）的用户自添加对象。
setAngle(float angle)	设定当前物理精灵的旋转角度。
createUI(LGraphics g)	此函数用以绘制当前物理精灵的相关图像到屏幕之上，如果需要进行特殊修改可以重载此函数。
setDrawImage(……)	设定指定图片或指定的图片路径为精灵显示用画面，但是此设置与初始化时的图片注入不同，此时 LGame 系统不会尝试自动获得图像边界。
setTag(Object tag)	设定任意对象到当前物理精灵中，用以标记此物理精灵或者缓存联动的数据。
setSpeed(……)	设定物理精灵移动速度。（实际调用 Body 的 setLinearVelocity）
dispose()	注销当前物理精灵。（会自动从 PhysicsScreen 中删除其对应的 Body）
getBitmap()	返回当前物理精灵所使用的图像。
getBody()	返回封装于当前物理精灵内部的 Body。
getTag()	获得当前物理精灵的内部标记。
setBitmapFilter(boolean bitmapFilter)	设定图像是否进行清晰化过滤，如果开启此项物理精灵在旋转时将不会存在模糊的边界（毛边），但显示速度也会较为明显的下降。
setPhysicsListener(PhysicsListener physicsListener)	设定一个物理精灵监听器，注入的监听器会与 PhysicsScreen 中的碰撞监听联动，可借此获得指定物理精灵的碰撞事件。

下面是一个关于 LGame 物理引擎的简单使用示例：

```
package org.loon.game.test.physics;

import org.loon.framework.android.game.core.graphics.device.LGraphics;
import org.loon.framework.android.game.core.timer.LTimerContext;
import org.loon.framework.android.game.physics.PhysicsObject;
import org.loon.framework.android.game.physics.PhysicsScreen;

import com.badlogic.gdx.math.Vector2;
import com.badlogic.gdx.physics.box2d.Body;
import com.badlogic.gdx.physics.box2d.BodyDef;
import com.badlogic.gdx.physics.box2d.FixtureDef;
import com.badlogic.gdx.physics.box2d.PolygonShape;
import com.badlogic.gdx.physics.box2d.BodyDef.BodyType;
import com.badlogic.gdx.physics.box2d.joints.RevoluteJointDef;

import android.view.MotionEvent;

public class Test1 extends PhysicsScreen {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public Test1() {
        // x 轴重力为 0,y 轴重力为 0.2,重力自然衰弱（不循环）
        super(0, 0.2f, false);
    }

    /**
     * 创建物理线桥
     *
     * @param imageName
     */
    private void createBridge(String fileName) {
        FixtureDef fixDef = new FixtureDef();
        PolygonShape poly = new PolygonShape();
        poly.setAsBox(6f, 2f);
        fixDef.density = 6f;
        fixDef.friction = 0.2f;
        fixDef.shape = poly;
```

```
        // 构建关节以联动桥板
        RevoluteJointDef rjd = new RevoluteJointDef();
        int numPlanks = 22;
        Body prevBody = null;
        Body firstBody = null;
        for (int i = 0; i < numPlanks; i++) {
            BodyDef bd = new BodyDef();
            if (i == 0 || i == numPlanks - 1) {
                bd.type = BodyType.StaticBody;
            }
            bd.position.set(3f + (15f * i), 320);
            Body body = world.createBody(bd);
            body.createFixture(fixDef);

            // 绑定指定路径下图片与 Body
            PhysicsObject o = bindTo(fileName, body);
            // 不允许图片旋转
            o.lockRotate = true;
            o.make();

            if (prevBody == null) {
                firstBody = body;
            } else {
                Vector2 anchor = new Vector2(2.4f + (15f * i), 320);
                rjd.initialize(prevBody, body, anchor);
                world.createJoint(rjd);
            }
            prevBody = body;
        }
        Vector2 anchor = new Vector2(2.4f + (15f * numPlanks), 320);
        // 初始化关节
        rjd.initialize(prevBody, firstBody, anchor);
        world.createJoint(rjd);
    }

    public void onLoad() {
        // 初始化完毕后创建物理桥
        createBridge("res/bridge.png");
        // 导入一张图片为物理精灵（LGame 可自动获得精灵图片的物理边界）
        PhysicsObject o = bindTo("res/a4.png", BodyType.DynamicBody, 50, 0);
        o.density = 2.0f;
        o.friction = 0.2f;
        o.make();
    }
```

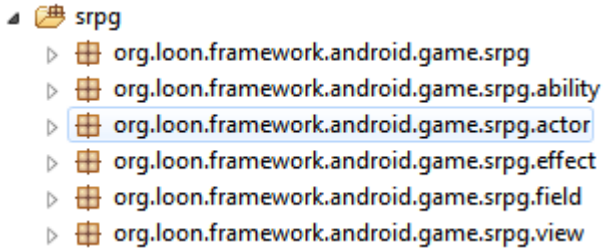
```
    public void paint(LGraphics g) {

    }

    public void update(LTimerContext t) {

    }

    public void onDown(MotionEvent e) {
        // 获得指定位置,且标记为"Ball"的物理精灵
        PhysicsObject ball = find(getTouchX(), getTouchY(), "Ball");
        if (ball != null) {
            // 存在则删除
            removeObject(ball);
        } else {
            // 添加一个圆形的物理精灵,初始位置在触摸屏点击区域，大小为 64x64
            ball = bindTo(Circle, BodyType.DynamicBody, getTouchX(),
                    getTouchY(), 64, 64);
            // 标记为 Ball（便于查找）
            ball.setTag("Ball");
            // 设定图片(请注意，当为固定形状注入图片时，图片大小必须为设定大小，
            // 否则会发生图片位置与物理位置不匹配的情况)
            ball.setDrawImage("res/ball.png");
            ball.setBitmapFilter(true);
            ball.density = 2.0f;
            ball.friction = 0.2f;
            ball.make();

        }
    }

    public void onMove(MotionEvent e) {

    }

    public void onUp(MotionEvent e) {

    }
}
```

运行效果如下所示：

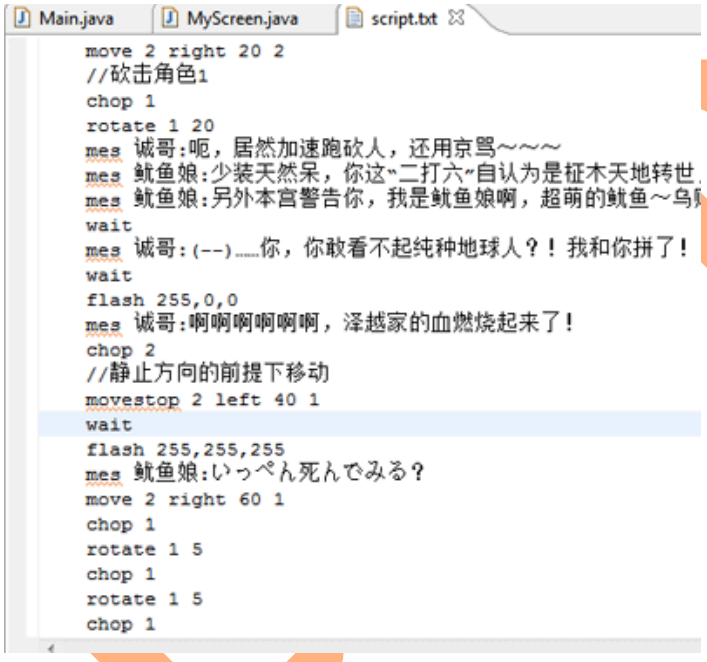
PS：即便构建方式完全一致，LGame-JavaSE 版的物理引擎运算效率也要 2、3 倍于 Android 真机之上，而在 Android 虚拟机运行时自然更低（基本上动态的物理精灵越多（下图的线桥也是动态精灵，只是在模拟器上没敢开线桥的图片旋转），运行速度就会越低，反之在全是静态精灵的前提下，显示速度会无限接近 60FPS（默认最大值））。

关于 SRPG 扩展包：

SRPG 类型游戏开发扩展包结构如下所示：



在设计上，SRPG 模块剧情模式目前仅支持硬编码及不完整的脚本设定（原本加入了第三方的 Lua 类库，但效果不好，另外小弟也打算稍候扩展 Command 类，所以暂时取消），不过已经足够满足大多数剧情交互需求甚至单纯 R 剧的演绎了。



为什么要在标准框架以外构建游戏扩展开发包

一直以来，我之所以将 LGame 版本号上升的很慢，其真正目的，原本就是为了在 1.0 版发布前完成 AVG、SLG（SRPG）、RPG、STG、ACT、PUZ、FTG、RTS 等八大扩展包，以求为用户提供一套完整的 2D 游戏解决方案，而不是一套单纯的渲染引擎或者工具集合。坦率的讲，LGame 本就是为了充当 2D 游戏领域的 Spring 而存在的。

至于使用扩展包所带来的具体好处，小弟粗略总结有如下两点，一曰“立象以尽意”，二曰“致一而不乱”。

一、立象以尽意

通常来说，越是复杂的游戏类型，越难以轻易的设计与开发出来，比如很多初学者在掌握了贪食蛇、俄罗斯方块之类简单游戏的制作方法后，便会自认为已初窥游戏开发门径，甚至意气风发的跳出来“指点江山”。可如果你想打击一下此人的锐气，也非常简单，只要让他在不使用 RMXP、RMVX、Sim RPG Maker 等业余游戏制作工具的前提下（PS：小弟并非歧视这些工具，毕竟有 amaranthia(http://www.amaranthia.com)这样收费的 RM 游戏网站存在，不过即便是 amaranthia 上出名的游戏，要移植到 iPhone 或 Android 等平台实现高盈利，也大多聘请专业程序员移植，而非原作者自行搞定），去独立制作一款水准近似上世纪九十年代早期如《运镖天下》、《炎龙骑士团》、《仙剑奇侠传》“在当时”效果的游戏，甚至让他去再现九十年代后期如《血狮》、《江湖》之类“令人疯狂的神作”，都可能让他信心顿失，原形毕露。

但请注意，小弟这里并不是说《炎龙骑士团》就比贪食蛇复杂了多少，《仙剑奇侠传》又比俄罗斯方块难在了何处，因为无论多么复杂的系统，都是由一个个简单到“1+1”程度的基础部分组合而成的。或者说，难道《炎龙骑士团》中需要碰撞算法，贪食蛇就不需要吗？难道《仙剑奇侠传》中需要事件触发，俄罗斯方块的消除方块步骤就不算事件吗？其实差别只在于，很多人都知道俄罗斯方块该怎么编码，而那些人却未必知道《炎龙骑士团》该怎样编码。

事实上，即便对很多首次尝试制作俄罗斯方块等级游戏的初学者而言，大多也不能一蹴而就的独立完成编程，他们每每会从四处找寻现成的源码“参考”开始，再经过几次编译失败或者论坛请教的“痛苦”过程，最终才“恍然大悟”的设计出那些“完全自主开发”的大作。然而，我们就事论事，其中真正能够“闭门造车，出门合辙”，全凭自悟写出相关算法者，恐怕也只有少数具备天才、鬼才、偏才一类天赋者才能做到。

反向例证，如果小弟能够在一夕之间让所有俄罗斯方块算法全部从网络以及教材中消失，并且让所有了解这种算法的活人全部闭嘴（乃至灭口），试问又会有多少后来人，能够自行参悟出俄罗斯方块的算法呢？凭心而论，恐怕不多吧。又好像说，如果在未来的“第三次世界大战”中毁灭了现存的全部人类科技成果，那么即便人类没有因战争灭绝，我们的子孙后代又能凭空在短时间内重建整个现代人类文明吗？哼哼，恐怕到时能不用石头与棒子进行“第四次世界大战”，就已经是最好的结局了。

所以，假设大家都生活在一个并不存在任何“俄罗斯方块源码或算法”，甚至根本没有存在过“俄罗斯方块”的世界里。这看上去简单已极的俄罗斯方块，又会有几个人能凭空创造出来呢？——凤毛麟角，恐怕都不足以说明其稀少程度。

如果您基本认同上述所言，那么，试问如果一个人压根就没有见过《炎龙骑士团》、《仙剑奇侠传》之类游戏究竟是怎么写成的，您又怎么可能要求在这个天才与白痴都仅占极少数的世界上，占绝大多数的、智商中等的游戏初学者们，能够拥有凭空完成同类游戏的能力呢？

从世俗的眼光来看，游戏开发初学者与游戏开发高手间的差距仿佛在于编程水平，仿佛“高手”就多么大牛，“新手”就多么小白。但归根结底，原来也不过是对于“前人经验”的掌握程度不同罢了。更极端的说，其症结在于贪食蛇、俄罗斯方块之类源码随处可见，而《炎龙骑士团》、《仙剑奇侠传》的源码却远没普及到人手一份……

我们中国人喜用“阳春白雪，下里巴人”来比喻某些曲高和寡的情形。但仔细想想，简单的“下里巴人”是歌曲（其实是两大部分），复杂的“阳春白雪”也不外是歌曲吧（同样两大部分）？歌曲是什么，不就是一种特殊频率的声音嘛；而声音是由什么决定的，大家都知道“高强长色”(音高、音强、音长、音色)这些概念。只要找准节奏，鹩哥、鹦鹉都能学人说话，更何况是人学人。就算“阳春白雪”有千句，“下里巴人”仅一行，也无外是歌词曲调罢了。如果词多，难道不能用笔记吗？如果音高，难道不能用低频唱吗？如果意深，难道不能人云亦云的模仿吗？能学会“下里巴人”，怎么可能就搞不定“阳春白雪”呢？

曹家老二教导我们说“文本同而末异”，孔家老二教导我们说“吾道一以贯之”。此刻，如果将那些简单的游戏类型比作“下里巴人”，那些复杂的游戏类型比作“阳春白雪”的话，就势必可以推论出，“能为下里巴人者，亦能为阳春白雪”这一浅显易懂的道理。说白了，能唱“下里巴人”的人，首先就已经拥有了最基础的歌唱能力，大约也算得智力正常（总不可能楚国上千人齐唱智障歌曲……），当然也能学唱“阳春白雪”，或许唱起来存在“好坏”的分野，却绝对不存在“能否”的问题，这道理非常浅显。

所以，绝大多数在游戏开发领域还唱着“下里巴人”调调的初学者，其真正需要的仅仅是一个会唱“阳春白雪”的人，教他“阳春白雪”到底该怎么唱法，乃至让他听一遍完整的“阳春白雪”演唱流程到底如何而已。这并非什么玄而又玄的大道理，不过是生活常识罢了。而目前中国游戏业，尤其是 Java 游戏领域的最大症结在于，根本没人教初学者该如何演唱“阳春白雪”，他举目所及又都是“下里巴人”之辈，初学者们当然也只好无限期的充当“下里巴人表演者”了。

古人为什么会说“言不尽意、立象以尽意”？归结其根本，就是怕后人无法理解前人的想法，所以才要“立象”，立个标杆给后人看看，后人才好有个参考，才知道该怎么办，不该怎么办，该怎么搞，不能怎么搞。能举一反三，能达到“得意而忘象”固然是好，如果没有那种天资，至少能够有样学样，也就“虽不中，亦不远，不为谬矣”了。

小弟在这里坦率的讲，只要掌握了适当的源码与算法，对绝大多数智力正常的游戏开发者——哪怕是初学者而言，无论开发任何游戏也至多存在时间问题，而决不会有能力问题。

如今，小弟提供细分的游戏扩展模块，首要目的就是“立象”，立个简单可行的标准给大家使用，借用《西游记》中孙悟空吓唬小和尚的话，我是“打个棍样”出来看看。

二、致一而不乱

三国时精研玄学（特指哲学上的，而不是后来充斥宗教色彩的玄学）的王弼曾说：“夫众不能治众，治众者至寡者也；夫动不能制动，制天下之动者，贞夫一者也。故众之所以得咸存者，主必致一也；动之所以得咸运者，原必无二也。物无妄然，必由其理，统之有宗，会之有元，故繁而不乱，众而不惑。”笼统的说，王弼这段话的精要就在于“以寡驭众，以简解繁，致一而不乱”。

大家都知道，所谓游戏框架或者说引擎，首先是一个大马甲，作者是什么 API 都敢往上调用，其次就是一个大箩筐，作者是什么模块都敢往里封装。他们为了照顾绝大多数使用者，那些已有的或潜在的需求，图像、音频、视频、陀螺仪感应、地图、精灵、物理引擎等等什么都

敢有……还不够用？没问题，还缺什么自己说，能说出来就能做出来。您还别不服气，只要你能说出名，绝大多数引擎作者就真能搞做出个对应模块给你看看。别说做不出，做不了不是显自己没本事吗？那还敢出来混？没这两下子，出门你都不好意思和别人打招呼。

可这样一搞，全倒是真全，广也是真广，但无论是性能上抑或专业性上，就全部打了折扣。为什么？俗话说的好，“贪多嚼不烂”啊，游戏引擎不是大力丸，不能包治百病，一旦想要解决所有问题，就难免会顾此失彼，这个用户说这样方便，那个用户说那样麻烦，你需要这个，他需要那个，真是出门的恨天阴，卖伞的恨天晴。一旦陷入这种因为产生问题而去解决问题的怪圈当中，就肯定会把有限的精力投入到无限的问题处理中去，那就永无出头之日了。为什么经常有所谓精英抨击开源引擎不够专业？其实他们真正看不上眼的，就是这种“万能特性”才对呀。

用户的需求，当然不能不满足，但也不能为了满足用户需求而无限度的扩展框架造成恶性循环。可这中间的矛盾该怎么解决呢？其实也好办，答案就是王弼所说的“得咸运者，原必无二”。

都说众口难调，但为什么再多人去饭馆吃饭，饭馆他也不怕众口难调啊？您什么时候去饭馆吃饭听说过：“先生，实在是众口难调，因此我们一次就管一个人做饭，队都排满了，您明年请早”的说法？原因很简单，饭馆按菜谱点菜，客人来饭馆不是想吃什么吃什么，而是菜谱有什么才能吃什么。客户的需求是无限的，但需求的种类是有限的，您口味再怪，也无非是“酸甜苦辣咸”这五味自由搭配嘛。连做饭的都懂这个道理，我们这些每天除了“计”就是“算”，除了“算”就是“计”的主怎么可能不懂呢？

当然，比喻是这么比喻，小弟可没打算直接“卖”游戏成品给用户，比较来看的话，其实使用引擎的用户才是“开饭馆”的，你们才负责“卖饭菜”给游戏玩家，而引擎作者不过是个原料供应商，存在的意义仅在给用户节省个种菜养猪的时间。不过，小弟虽然不能“卖”你“成品”，可谁又说小弟连个“半成品”也不能“卖”你呢？

只要不玩穿越，现存的所有 javaer 恐怕都知道 Spring，都知道一个 Spring 就能搞定几乎所有的 Java 企业级开发需求（不过某些复杂业务也需要其它组件支持）。可难道你对着 Spring 的 jar 大叫“芝麻开门”，它就会自行实现并构建出你所需要的完整代码了吗？肯定不是。难道是所有用户都不停的向 Rod Johnson 提要求，他就不停地实现用户要求了吗？肯定也不是（各种意义上都不是|||）。事实是，Spring 也不过是提供了一系列标准化的组件，为近乎所有企业级需求都提供了基础实现模块（或者第三方库的更简易封装），而任凭用户自由选取这些现成模块二次组合罢了。

既然同属 javaer，既然有前人的成功经验，即使领域略有差别，小弟却一样可以照葫芦画瓢。我把 AVG、SLG、RPG、STG、ACT、PUZ、FTG、RTS 这些常见游戏模式各做一个标准封装，怎么着你也能用上一个吧？一切按标准走，这样就你也不乱，我也不乱了。

没错，小弟提供的游戏扩展模块，另一个意义就是充当一个“游戏半成品”或者说是一个“特定游戏类型的开发标准”，只要用户想开发的游戏与该扩展包属同一类型（综合类型混用即可），那么按照扩展包的套路去做，甭管您会不会“做饭”吧，至少也能炒个“菜”出来。它的存在，能够让开发《炎龙骑士团》变得像开发贪食蛇一样简单，让制作《仙剑奇侠传》变

得比制作俄罗斯方块还要轻松。

最基本的 SRPGScreen 类构造方法：

```
public class SRPGTest1 extends SRPGScreen{

    public SRPGTest1(){
        //游戏地图（二维数组），瓦片宽，瓦片高
        //PS：构造方法较多，此处仅列出一种
        super("assets/map.txt",48,48);
    }

    //SRPGScreen 核心围绕一个名为 mainProcess 的函数运行，该函数内部将不停循环，
    //每循环一次代表经过一个回合，而 startProcess 将在执行 mainProcess 前获得执行，
    //如果 return 为 true，则 startProcess 不会停止，也就不会执行 mainProcess
    protected boolean startProcess() {
        return false;
    }

    //SRPGScreen 核心围绕一个名为 mainProcess 的函数运行，该函数每循环一次
    //代表经过一个回合，而 endProcess 将在 mainProcess 循环完全结束后获得执行，
    //如果 return 为 true，则 endProcess 不会停止，当前 Screen 也就不会正常关闭
    protected boolean endProcess() {
        return false;
    }

    //当游戏刚刚启动时，将自动调用此函数，可遇此处添加游戏角色资料
    //它的优先级比所有含有【Process】字样的函数都高
    protected void initActorConfig(SRPGActors actors) {
    }

    //当游戏刚刚启动时，将自动调用此函数，可遇此处添加地图瓦片资料（即设定地形参数）
    //它的优先级比所有含有【Process】字样的函数都高
    protected void initFieldElementConfig(SRPGFieldElements elements) {
    }

    //当游戏刚刚启动时，将自动调用此函数，可遇此处变更地图显示形式。
    //可以设定的形式有三种：
    //1、以小图贴成地图
    //2、直接使用完整图片作为地图
    //3、底层为完整图片，上面覆盖部分小图贴图
    //它的优先级比所有含有【Process】字样的函数都高
```

```
    protected void initMapConfig(SRPGField field) {
    }

    //当游戏刚刚启动时，将自动调用此函数，可于此处设定游戏角色分组状态（敌我归类）
    protected void initTeamConfig(SRPGTeams team) {
    }

    //获得当前选中的游戏角色，以及对应的地图位置
    public void onClickActor(SRPGActor actor, int x, int y) {
    }

    //获得当前选中的游戏地形，以及对应的地图位置
    public void onClickField(SRPGFieldElement element, int x, int y) {
    }

    //触屏按下
    public void onDown(Touch e) {
    }

    //触屏移动
    public void onMove(Touch e) {
    }

    //触屏放开
    public void onUp(Touch e) {
    }

    //在首次执行 mainProcess 函数时将触发此函数
    protected void processInitialize() {
    }

    //当游戏回合改变后将触发此函数
    protected void processChangePhaseAfter() {
    }

    //当游戏回合改变前将触发此函数
    protected void processChangePhaseBefore() {
    }

    //当发生攻击事件时，在角色进行攻击后将触发此函数，返回正在攻击的角色类及其 ID
    protected void processAttackAfter(int index, SRPGActor actor) {
    }

    //当发生攻击事件时，在角色即将攻击前将触发此函数，返回正在攻击的角色类及其 ID
```

```
    protected void processAttackBefore(int index, SRPGActor actor) {
    }

    //当发生攻击事件后，会将储存伤害参数值的 SRPGDamageData，以及攻击与防御方 ID 传递到此
    protected void processDamageInputAfter(SRPGDamageData damagedata, int atk,
            int def) {
    }

    //当发生攻击事件前，会将储存伤害参数值的 SRPGDamageData，以及攻击与防御方 ID 传递到此
    protected void processDamageInputBefore(SRPGDamageData damagedata, int atk,
            int def) {
    }

    //当角色临近死亡时，在其判定死亡后，会将其 ID 和对应类传递至此
    protected void processDeadActorAfter(int index, SRPGActor actor) {
    }

    //当角色临近死亡时，在其判定死亡前，会将其 ID 和对应类传递至此
    protected void processDeadActorBefore(int index, SRPGActor actor) {
    }

    //当角色升级后，将触发此函数，并将其 ID 和对应类传递至此
    public void processLevelUpAfter(int index, SRPGActor actor) {
    }

    //当角色升级前，将触发此函数，并将其 ID 和对应类传递至此
    public void processLevelUpBefore(int index, SRPGActor actor) {
    }

}
```

一些使用上的要点：

1、所有 SRPG 角色都由 SRPGStatus 与 SRPGActor 两大部分组成，其中 SRPGStatus 为角色的技能、属性、脸谱、是否主角、AI 类型，是否由电脑操作，角色分组等参数设置，而 SRPGActor 类决定角色在游戏大地图上的显示画面，坐标位置等参数。

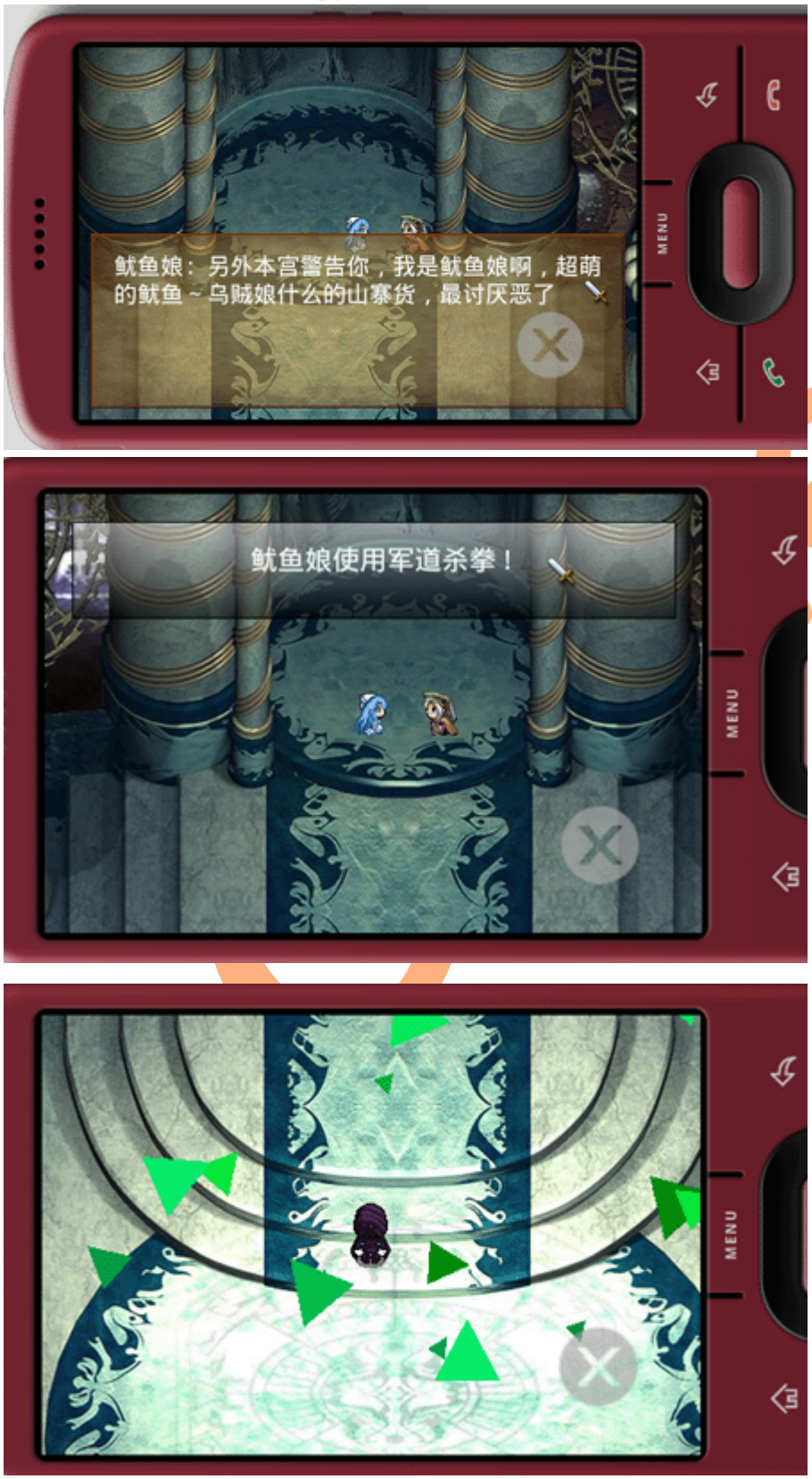
2、由于 SRPGScreen 采用双线程操作，所以如果要在某些循环中停止 LGame 绘图主线程，建议直接使用封装好的 waitFrame 或 waitTime 方法。

3、如果您不想进入战斗，而只是对当前 SRPGScreen 进行地图浏览，可以在初始设定时使用 setBattleMode(false)来阻止战场主循环运行。

4、SRPG 模块的常规技能设定通过 SRPGAbilityFactory 类，常规角色（职业）设定通过 SRPGActorFactory 类，两者中各有一组默认设置可供参考。

5、虽然目前提供的脚本功能并不完全，但您可以通过重载 onAvgViewNext 函数进行扩充。

以下为示例画面（LGame-Srpg 模块默认效果）



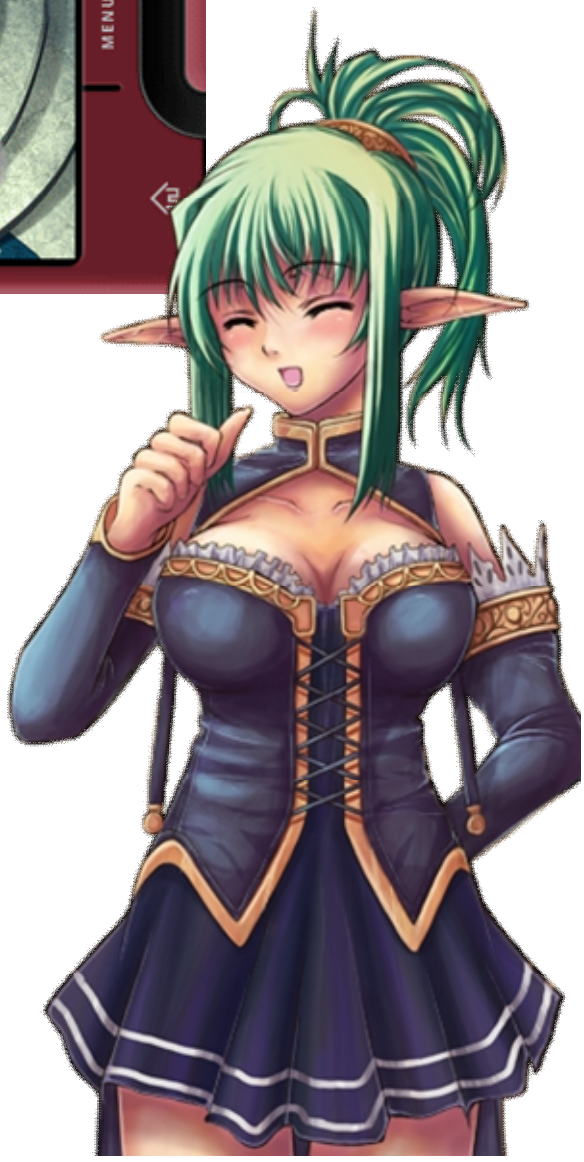

战斗方式
道具
转换方向
属性
行动结束
MENU


斩击 1-1
军道杀拳 1-2 100
曙光女神之宽恕 1-2 100
吸星大法 1-1 1
休息 0-0
MENU


魷鱼娘
火星来的灌水女王
Lv 95 Exp 0
HP 1026 / 1026
MP 488 / 488
Ability Length Main Leader
斩击 1-1 0 STR 967
军道杀拳 1-2 1 100 DEX 706
曙光女神之宽恕 1-2 2 100 VIT 756
吸星大法 1-1 0 1 AGI 516
休息 0-0 0 MAG 10
MOV 3
转换方向
属性
行动结束
MENU

LGame-Android-0.3.0 版的函数说明到此结束，未涉及的功能及函数会在未来文档中陆续补全，敬请期待。

下面开始，将是涉及 LGame 引擎使用配置的简略说明。



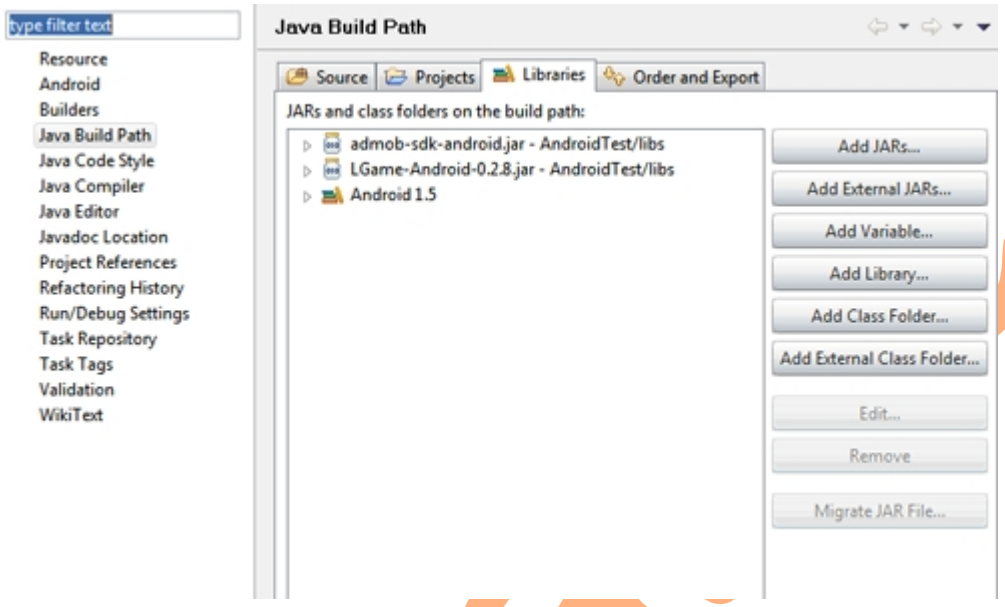
关于 LGame 引擎使用的基本配置

（1） 如何导入 LGame 引擎

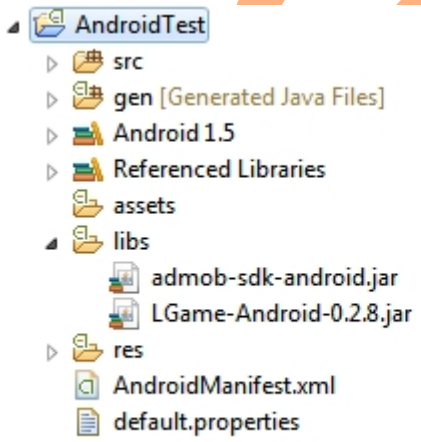
在 Android 开发环境中使用 LGame 非常之简单，只需要

我们将 jar（LGame-Android 以及 Admob 的相关 jar）放置在 Libs 文件夹下，并且配置相关 Lib 库即可。而对于需要使用 so 文件的部分（比如物理引擎），只要导入相关 jar，并将本引擎提供的 armeabi 或 armeabi-v7a（armeabi-v7a 适用于 Android 2.2 及更高系统,而 armeabi 可以跑全环境，不过针对 armeabi 的 so 文件体积较大）文件夹拷贝到 libs 文件夹下即可，Android 系统会自动识别并加载调用。

具体如下图所示：



当成功导入相关 jar 后，当前项目将呈现出类似下图的画面：



此时，我们就可以使用 LGame 引擎开始 Android 史上最为轻松的游戏开发之旅了。

（2） AndroidManifest 文件应该如何描述

众所周知，Android 的执行文件格式为 APK，而这些 APK 的启动参数又被硬性规定必须构建于 AndroidManifest.xml 当中。所以使用 LGame 引擎，也不可免俗的必须配置 AndroidManifest.xml 文件，万幸的是，只需要很少的操作就可以做到了。

一、创建继承自 LGameAndroid2DActivity 的启动类

假如我们要创建一个文件名为【Main】的启动用类，仅仅需要做如下简单操作：

```
//继承 LGameAndroid2DActivity
public class Main extends LGameAndroid2DActivity {
    //实例化抽象函数 onMain
    public void onMain() {
        //屏幕横屏、有广告且居于屏幕右方、Admob ID 为 XXXX，广告刷新速度为 60 秒
        //this.initialization(true,LAD.RIGHT, "Admob ID",60);
        //屏幕横屏
        this.initialization(true);
        //注入指定的 Screen 窗体实例
        this.setScreen(new Test());
        //设定 FPS 为每秒 30
        this.setFPS(30);
        //不显示 logo
        this.setShowLogo(false);
        //显示 FPS
        this.setShowFPS(true);
        //显示 Screen 窗体
        this.showScreen();
    }

}
```

二、配置 AndroidManifest.xml

此时，我们只要对 xml 进行近似的配置（并非一定要全部照抄，最关键的是在配置中必须要有 android:configChanges="orientation|keyboardHidden"，否则将无法强行横屏或竖屏），就可以开始 LGame 之旅了。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="org.loon.Main"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
```

```
        <activity android:name=".Main"
                  android:configChanges="orientation|keyboardHidden"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="3" />
    <uses-permission android:name="android.permission.INTERNET"/>
</manifest>
```

（3） 如何使用各种 Screen

一、Screen

Screen 是 LGame 引擎中最基本的窗体模式，采取自动刷新，刷新频率由 setFPS 方法决定。

```
import org.loon.framework.android.game.core.graphics.Screen;

import android.view.KeyEvent;
import android.view.MotionEvent;

public class ScreenExample extends Screen{

    // draw中LGraphics会根据设定的FPS自动刷新，使用上与标准的J2SE Graphics以及
J2MEGraphics接口没有区别（API为二者的综合）
    public void draw(LGraphics g) {
    }

    //键盘按下
    public boolean onKeyDown(int keyCode, KeyEvent e) {
        return true;
    }

    //键盘放开
    public boolean onKeyUp(int keyCode, KeyEvent e) {
        return true;
    }

    //触摸屏按下
    public boolean onTouchDown(MotionEvent e) {
```

```
        return true;
    }

    //手指在触摸屏上移动
    public boolean onTouchMove(MotionEvent e) {
        return true;
    }

    //触摸屏放开
    public boolean onTouchUp(MotionEvent e) {
        return true;
    }

}
```

另外，在Screen中还有一个比较重要的alter函数，我们可以通过重载alter函数实现最简单的定时操作，譬如:

```
    //设定计时器，每隔1秒允许执行一次
    LTimer timer=new LTimer(LSystem.SECOND);

    //重载alter函数
    public void alter(LTimerContext context){
         if(timer.action(context.getTimeSinceLastUpdate())){

         }
    }
```

二、CanvasScreen

CanvasScreen 模拟 J2ME 中 Canvas 而成（也混合了 GameCanvas 的 API），在 repaint 前不会自动刷新，相关函数在 J2SE 及 Android 版中完全一致。

```
import org.loon.framework.game.simple.core.graphics.CanvasScreen;
import org.loon.framework.game.simple.core.graphics.device.LGraphics;

public class CanvasScreenExample extends CanvasScreen {

    public CanvasScreenExample(){
        //与ThreadScreen相同，CanvaScreen允许改变游戏图像大小。
        //以下参数分为别原始的图像宽与高（300x450），要求显示的宽与高(320x480)
        super(300,450,320,480);
    }
```

```
    // CanvasScreen绘图器，与标准Screen中draw函数的区别在于paint的修改仅在
repaint时生效，而且paint不会自动清除原有的图像内容。
    public void paint(LGraphics g) {

    }

    // 键盘按下
    public void keyPressed(int keyCode) {

    }

    // 键盘放开
    public void keyReleased(int keyCode) {

    }

    // 触摸屏或鼠标移动
    public void pointerMove(double x, double y) {

    }

    // 触摸屏或鼠标按下
    public void pointerPressed(double x, double y) {

    }

    // 触摸屏或鼠标放开
    public void pointerReleased(double x, double y) {

    }

}
```

三、AVGScreen

AVGScreen 是一个非常特殊的 Screen，只有在注入脚本后才能发挥作用。他直接继承自 CanvasScreen，用于进行 AVG 类游戏开发或者制作较为复杂的过场效果（也可以配合 LGame 组件制作游戏商店界面等）。AVGScreen 默认采取自动画面刷新模式，不过也允许手动调用 repaint。

```
public class AVGScript extends AVGScreen {
```

```
    /**
     * 使用指定脚本，指定路径中图片为对话框
     *
     * @param initscript
     * @param initdialog
     */
    public AVGScript(String initscript, String initdialog) {
        super(initscript, initdialog);
    }

    /**
     * 使用指定脚本，指定对话框图片
     *
     * @param initscript
     * @param img
     */
    public AVGScript(String initscript, Image img) {
        super(initscript, img);
    }

    /**
     * 使用指定脚本，默认对话框
     *
     * @param initscript
     */
    public AVGScript(String initscript) {
        super(initscript);
    }

    /**
     * AVG底层绘图接口，允许直接在此绘制您所需要的画面(即使不使用底层绘图， 您也可以通过add方式增加组件或精灵到指定位置)
     */
    public void drawScreen(LGraphics g) {

    }

    /**
     * 仅在初始化时起作用，设定脚本命令参数，可于此处预设游戏中变量供脚本读取
     */
    public void initCommandConfig(Command command) {

    }
```

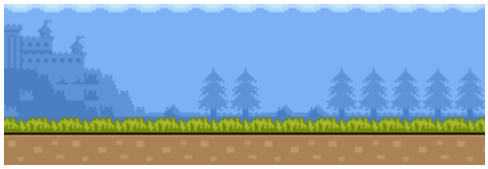
```
    /**
     * 仅在初始化时起作用，设定信息框参数，不填此项时，对应组件默认置于屏幕下方，自适应信息框图片
     */
    public void initMessageConfig(LMessage message) {

    }

    /**
     * 仅在初始化时起作用，设定选择框参数，不填此项时，对应组件默认置于屏幕下方，自适应选择框图片
     */
    public void initSelectConfig(LSelect select) {

    }

    /**
     * 当执行AVG脚本时触发此项，message中数据为脚本信息，若返回false， 则当前脚本将被此函数截取（此时默认解释器无效），用户可以自行解析
     * 该行脚本数据。
     */
    public boolean nextScript(String message) {
        return true;
    }

    /**
     * 当画面中出现选择项，并且选中时触发此函数，message中数据为[select]命令信息，type为选中的选择项，索引由0开始
     */
    public void onSelect(String message, int type) {
    }

    /**
     * 当脚本执行[exit]命令时将调用此参数，可在此执行setScreen之类命令离开AVG窗体
     */
    public void onExit() {
    }

}
```

下面将提供一个非常简单的 CanvasScreen 示例，示例内容为在窗体上自定义一群跑动的精灵，并将背景及窗体扩充到我们需要的大小。

//精灵出现时画面

//精灵移动时画面

//游戏背景窗体



请注意，为了演示图像扩大机制，实例所用的精灵以及游戏背景图像都非常之小，实际开发中并非一定要使用图像扩充，更不是只能使用此比例的图像。

1、Sword.java

Sword 是一个自定义的精灵类，用以显示上图所示的“骷髅剑士”。PS：在不重新实现 ISprite 的基础上，我们也可以直接使用现成的 Sprite 类来完成此例。但是，关于动作细节部分就只能在精类外部设定或者重载 Sprite 的 update 函数来实现，这样和重新实现 ISprite 没有太大区别。

```
import org.loon.framework.game.simple.action.map.RectBox;
import org.loon.framework.game.simple.action.sprite.Animation;
import org.loon.framework.game.simple.action.sprite.ISprite;
import org.loon.framework.game.simple.action.sprite.SpriteImage;
import org.loon.framework.game.simple.core.LObject;
import org.loon.framework.game.simple.core.LSystem;
import org.loon.framework.game.simple.core.graphics.device.LGraphics;
import org.loon.framework.game.simple.core.timer.LTimer;

public class Sword extends LObject implements ISprite {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    // 精灵移动范围的最大宽度（即背景图实际宽，为240）
    private static final int WIDTH = 240;
```

```
    private boolean init, right, flag, visible;

    private int randX;

    private Animation animation;

    private LTimer time;

    private static final String kName1 = "res/s1.png", kName2 =
"res/s2.png";

    public Sword() {
        // 加载怪物剑士的“出土”动画（以下参数为图像所在地址、图像宽、图像高、播放间
隔）
        this.animation = Animation.getDefaultAnimation(kName1, 18, 18,
150);
        // 设定计时器间隔为10豪秒
        this.time = new LTimer(10);
        // 设定精灵初始位置（x轴随机，y轴48（立于相对于背景视觉的“地面”上））
        this.setLocation(LSystem.random.nextInt(WIDTH), 48);
        // 随机决定精灵剑士向左或向右冲锋
        this.right = LSystem.random.nextInt(2) == 0 ? true : false;
        // 随机决定精灵剑士以匀速或两倍速前进
        this.flag = LSystem.random.nextInt(2) == 0 ? true : false;
        // 随机决定精灵剑士的“冲锋”停止点
        this.randX = LSystem.random.nextInt(70);
        // 当前精灵可见
        this.visible = true;
    }

    // 设定精灵绘图器内容
    public void createUI(LGraphics g) {
        if (animation != null) {
            // PS:Android版中没有提供序列化保存SpriteImage的方法，直接getImage
即可，而没有serializablelImage。
            // 反转图像
            if (right) {

    g.drawMirrorImage(animation.getSpriteImage().serializablelImage
                      .getImage(), x(), y());
                // 正向
            } else {

    g.drawImage(animation.getSpriteImage().serializablelImage
```

```
                    .getImage(), x(), y());
            }
        }
    }

    // 当前精灵宽
    public int getWidth() {
        SpriteImage si = animation.getSpriteImage();
        if (si == null) {
            return -1;
        }
        return si.getWidth();
    }

    // 当前精灵高
    public int getHeight() {
        SpriteImage si = animation.getSpriteImage();
        if (si == null) {
            return -1;
        }
        return si.getHeight();
    }

    // 精灵计时器，用以在timer满足条件时变更精灵样式
    public void update(long timer) {
        if (time.action(timer)) {
            animation.update(timer);
            // 当动画没有播放时
            if (!animation.isRunning()) {
                // 初始化动画精灵（以下参数为图像所在地址、图像宽、图像高、播放间隔）
                animation = Animation.getDefaultAnimation(kName2, 26, 18,
50);
            } else if (animation.isRunning() && init) {
                // 向左
                if (!right && (x() + randX) > 0) {
                    if (flag) {
                        // 精灵向左移动
                        move_left();
                    } else {
                        // 精灵向左移动（速度x2）
                        move_left(2);
                    }
                    if ((x() - randX) <= 0) {
                        right = true;
```

```
                }
                // 向右
            } else if (right
                    && (x() + animation.getSpriteImage().getWidth() -
randX) <= WIDTH) {
                if (flag) {
                    // 精灵向右移动（速度x2）
                    move_right(2);
                } else {
                    // 精灵向右移动
                    move_right();
                }
                // 当达到屏幕边缘时，改变移动方向
                if ((x() + animation.getSpriteImage().getWidth() +
randX) >= WIDTH) {
                    right = false;
                }
            }
        }
        // 当精灵动画结束时，设定init=true(精灵动画播放完毕)
        if (!init
                && animation.getTotalFrames() - 1 == animation
                        .getCurrentFrameIndex()) {
            animation.setRunning(false);
            init = true;
        }
    }
}

// 透明度
public float getAlpha() {
    return 0;
}

// 碰撞盒
public RectBox getCollisionBox() {
    return new RectBox(Math.round(x()), Math.round(y()), getWidth(),
            getHeight());
}

// 显示状态
public boolean isVisible() {
    return visible;
}
```

```
    // 是否显示精灵
    public void setVisible(boolean visible) {
        this.visible = visible;
    }

}
```

2、CanvasScreenExample.java

设定一个CanvasScreenExample.java，用以继承CanvasScreen。

```
import org.loon.framework.game.simple.GameScene;
import org.loon.framework.game.simple.action.sprite.Sprites;
import org.loon.framework.game.simple.core.graphics.CanvasScreen;
import org.loon.framework.game.simple.core.graphics.Deploy;
import org.loon.framework.game.simple.core.graphics.LImage;
import org.loon.framework.game.simple.core.graphics.device.LGraphics;


public class CanvasScreenExample extends CanvasScreen implements Runnable
{

    // 设定精灵组，显示范围为宽240，高80（此为实际背景图大小）
    private Sprites sprs = new Sprites(240, 80);

    // 设定背景图像
    private LImage background = LImage.createImage("res/background.png");

    // 是否允许线程运行
    private boolean running;

    // 刷新间隔
    private long speed = 30;

    public CanvasScreenExample() {
        // 重载游戏画面大小，将240x80的实际游戏图像大小，扩大为480x160的显示大小
        super(240, 80, 480, 160);
        // 创建五个骷髅剑士的精灵
        Sword[] sw = new Sword[5];
        //循环载入
        for (int i = 0; i < sw.length; i++) {
            // 将精灵载入精灵管理器
```

```
            sprs.add(sw[i] = new Sword());
        }
        // 允许线程运行
        this.running = true;
        // 启动线程
        this.callEvent(new Thread(this));

    }

    // 当切换不同的Screen，或者游戏结束时，会触发此函数
    public void dispose() {
        this.running = false;
    }

    // 在绘图器中绘制背景图以及相关精灵
    public void paint(LGraphics g) {
        // 绘制背景图像到位置(0,0)
        g.drawImage(background, 0, 0);
        // 绘图精灵组中所有精灵(默认位置(0,0)，也允许通过setLocation或
setViewWindow变更显示位置及显示范围)
        sprs.createUI(g);
    }

    public void run() {
        while (running) {
            // 刷新游戏画面
            this.repaint();
            // 间隔30豪秒
            this.pause(speed);
            // 刷新精灵组
            sprs.update(speed);
        }
    }

    public void keyPressed(int keyCode) {

    }

    public void keyReleased(int keyCode) {

    }

    public void pointerMove(double x, double y) {
```

```
    }

    public void pointerPressed(double x, double y) {

    }

    public void pointerReleased(double x, double y) {

    }

}
```

当我们在手机中运行 CanvasScreenExample 时（setScreen(new CanvasScreenExample())），就可以得到游戏画面如下所示：

PS:在 CanvasScreen 中也允许完全照搬 J2ME 中 Canvas(GameCanvas)与相关精灵的构建方式(使用 sprite.j2me 包)，可以将任何 J2ME 游戏代码平移其中。

如果您问 LGame 是工具类吗？我会回答您：“是，也不是”；如果您问 LGame 是渲染引擎吗？我会回答您：“是，也不是”；如果您不厌其烦的来询问 LGame 是否一套完整的游戏引擎时，我还会回答您：“是，也不是”。

假如您仍然要追问这 LGame 究竟是什么“鬼东西”，小弟只好这样告诉您：其实，LGame 不过是一套完整的 2D 游戏解决方案，不过是一个 2D 游戏领域的“Spring 之流”罢了。