# 最后更新时间：2012-11-21

# core

core层主要对引擎的核心接口做了封装，同时提供了一些lua底层模拟面向对象的基础架构，也引入了一些基础结构。以下是各类说明。

## StateMachine

类说明：

StateMachine 提供了场景切换管理的全局控制。全局单键。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| StateMachine.getInstance | function() | 获取全局实例。 |
| StateMachine.releaseInstance | function() | 程序结束时调用，其实也可以不调用，反正都退出了 |
| StateMachine.changeState | function(self,state,style)  state – StateConfig中的枚举值  style – 切换模式 | 切换到下一状态，以前状态全部释放。 |
| StateMachine.pushState | function(self,state,style,popup)  参数同上 | 切换到下一状态，以前状态未释放，是否隐藏要视popup而定。下面状态切换步骤有叙述。 |
| StateMachine.popState | function(self,style) | 切换到上一保存状态，当前状态释放。 |
| StateMachine.registerStyles | function(key,func) | 注册自己的场景切换风格，key 为 上述style 的名字，func为一个函数，此函数会在场景切换是回调，参数如下：  newStateObj, --新状态对象  lastStateObj, -- 上一状态对象  doNeedReleaseLastStateObj  --是否需要释放上一状态 |

## State

类说明：

用于StateMachine的状态基类，每个用于StateMachine的state必须继承自State。而且每个重写的接口，必须调用基类相应接口。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| State.ctor | function(self) | 此为构造函数，状态的构造函数中不应该包含大量的资源加载，应该很快退出完成。 |
| State.load | function(self) | 此接口用于加载资源，加载完成后返回true。此函数会持续调用知道返回true，中间可返回false用做步进加载。 |
| State.run | function(self) | 此接口应该将界面显示出来。 |
| State.resume | function(self) | 在此接口中，应该加界面的持久动画启动。此接口之后正式进入新状态开始运行。 |
| State.pause | function(self) | 此接口应该做和resume相反的动作，停止动画。 |
| State.stop | function(self) | 此接口用于隐藏状态不应释放资源。run的逆操作。 |
| State.unload | function(self) | 此接口应该释放资源。load的逆操作。 |
| State.dtor | function(self) | 析构函数，彻底释放状态资源。 |

如果有两个状态，要从状态A 切换到状态B，调用顺序是这样的：

1. B ctor
2. A pause
3. B load 如果需要加载资源的话
4. B run
5. B resume
6. A stop
7. A unload
8. A dtor

其中7-8，如果是changeState或popState一定会调用。如果是pushState则会有不同，如果pushState最后一个参数popup是true，则都不执行，如果popup是false，则只执行7。

## System

类说明：

此类封装了各种系统查询，系统设置函数。主要是对引擎中提供的各种sys\_set\_\*，sys\_get\_\* 的封装。具体的参数解释可以查询BABE\_Kernel\_Reference，这里不解释。

注意：

最好用此类封装，而不要直接使用sys\_\*\_\* ，因为将来可能会在这一层做一些改动，而且这一层也可帮忙屏蔽引擎对接口的不兼容升级。

## Sound

类说明：

此类封装了声音部分接口。主要是对引擎中audio\_music\_\* 和 audio\_effect\_\*的封装。具体可参考BABE\_Kernel\_Reference。

注意：

同System。

## Socket

类说明：

此类封装了引擎提供的socket接口。不同于System和Sound，socket提供的不是静态接口，而是类接口。所以使用时需new一个socket对象，然后使用。

注意：

1. 由于引擎中只有room socket 和 hall socket 所以，socket的构造函数的参数只能为 “room” 或 “hall”。
2. 接口名定义如下规则：引擎中为socke\_\*\_\*，第一个\*为应该传入构造函数的room或hall，第二个\*为socket类中定义的函数名(\_全部去掉，相邻单词首字母大写)。
3. 接口参数与引擎接口相同。
4. 同上，尽量用此类，而非引擎接口。

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| Socket.setOnEvent | function(self,obj,func)  同普通回调参数 | 回调时参数为  func(obj,eventType, param); |

## Http

类说明：

此类封装了引擎提供的Http接口。

注意：

1. 构造函数和析构函数分别封装了http\_request\_create和http\_request\_destroy表新建和释放一个http请求。
2. 其他接口规则同socket。
3. Http.setEvent 可以设置这一http请求结束后的回调函数，附加参数会传回当前http对象。

## Dict

类说明：

此类封装了引擎提供的dict接口。dict用于构造数据字典，既可以用于临时保存东西到内存中，也可以用于可持久化保存、读取信息。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| dict.ctor | function(self,dictName) | 此为构造函数，创建一个字典，此时字典无实体，只是一个placeholder。 |
| dict.load | function(self) | 如果构造的字典是一个已经保存在硬盘中的结构，则需要调用此函数从硬盘中已存在的读取字典内容。 |
| dict.save | function(self) | 将字典保存在硬盘中。 |
| dict.delete | function(self) | 删除字典。 |
| dict.setInt | function(self,key,value) | 设置键值对，值必须是int。 |
| dict.setBoolean | function(self,key,value) | 设置键值对，值必须是bool型。 |
| dict.setDouble | function(self,key,value) | 设置键值对，值为double型。 |
| dict.setString | function(self,key,value) | 设置键值对，值为字符串型。 |
| dict.getInt | function(self,key,defaultValue) | 读取int值，如果提供的key不存在，则返回defaultValue（默认为0）。 |
| dict.getBoolean | function(self,key,defaultValue) | 读取bool值，如果提供的key不存在，则返回defaultValue（默认为false）。 |
| dict.getDouble | function(self,key,defaultValue) | 读取double值，如果提供的key不存在，则返回defaultValue（默认为0.0）。 |
| dict.getString | function(self,key) | 读取字符串，返回空。 |
| dict.dtor | function(self) | 析构函数 |

## Scene

类说明：

此类用于解析UIEditor生成的界面配置文件。

构造之后，会将整个场景解析出来，并加载进内存中，之后可以通过getRoot得到场景根节点进行操作。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| Scene.ctor | function(self,t)  t – 生成的lua文件的最外层table。 | 此为构造函数，生成场景。 |
| Scene.getRoot | function(self) | 得到生成场景的根节点。 |
| Scene.registLoadFunc | function(name,func) | 静态接口，对于自定义控件注册一个load函数。注册的函数应该返回一个节点对象。具体函数编写，可参数Scene.load\* 部分。 |
| Scene.dtor | function(self) | 析构函数，释放场景。 |

## GameString

类说明：

此类主要封装了游戏中字符串的操作函数。包括从字符串配置文件中加载字符串，以及字符串编码转化函数。

平台编码中win32中文比较特别，要区分是简体还是繁体，要配置这一属性应该在config.lua 中 加入LanguageConfig.isZhHant = true配置为繁体，否则为简体。

全部都是静态接口。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| GameString.setWin32Code | function(win32Code)  win32Code – “gb2312” or “gbk” | 设置win32下的中文编码方式，繁体用”gbk”,简体用”gb2312”，默认为”gb2312”。 此接口调用后会持续影响中文编码。 |
| GameString.load | function(filename)  filename 文本文件名 | 加载文本文件。 |
| GameString.get | function(key) | 根据文本文件中的key得到对应字符串，应该始终为utf8编码。 |
| GameString.convert | function(str) | 把一个UTF-8编码字符串转为平台编码 |
| GameString.convert2Platform | function(str,sourceCode)  str—字符串  sourceCode – 字符串编码 | 把str从特定编码转为平台编码。 |
| GameString.convert2UTF8 | function(str) | 把平台编码的字符串转为UTF-8编码。 |

## EventDispatcher

类说明：

本类提供一个消息分发中心。有两个方面的用途：

1. 引擎中本身提供了一些全局消息，如果用户需要监听这类消息就要到eventDispatch中注册监听，那么在引擎中消息分发的时候，响应的函数就可以得到回调。
2. 用户自己也可以注册一些消息类型用于低耦合通信，这时需要监听者注册监听，发送消息者通过dispatch分发消息。当发送者分发的时候，注册的回调可以得到调用。

用户注册自己的消息，应该保证全局唯一，不与其他消息重合，包括引擎消息。要注册自己的消息只需要将一个不同于引擎消息的值传入各个函数的event参数就可以了，此参数可以为任何可做table键类型的值。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| EventDispatcher.getInstance | function() | 静态接口，得到全局实例。 |
| EventDispatcher.releaseInstance | function() | 游戏结束时调用，释放实例。 |
| EventDispatcher.register | function(self,event,obj,func)  event – 事件类型  obj,func 同普通回调函数  Event = {  RawTouch = 1,  }; | 注册消息，调用此方法后，当事件出现时，回调函数会得到调用。调用参数为  func(obj,…);其中…为分发时传入的其他参数。  现在系统消息只有rawTouch，其… 和drawing中touch事件收到的值相同。 |
| EventDispatcher.unregister | function(self,event,obj,func)  参数同上 | 清除注册事件，调用此方法后不会再受到回调。 |

## Object

文件说明：

此文件中包含的并不是一个类，而是四个全局接口，用于模拟面向对象结构。

注意：

1. 只支持单继承。
2. 静态成员如需用做左值只能通过class.member的方式访问。
3. 调用delete方法后并不意味着彻底回收内存，需要在调用delete之后，再将对象置为nil，并且确保其他地方无引用，可保证释放。
4. 接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| new | function(class\_type,...)  class\_type -- 类  … -- 传入构造函数所需参数 | new会按照与C++相同的构造顺序，从基类到派生类逐步进化构造一个对象。 |
| delete | function(obj)  obj – 要释放的对象 | delete会按照与C++同样的方式，从派生类到基类逐步退化去释放一个对象。 |
| class | function(super,autoCallSuperCtor)  super –基类  autoCallSuperCtor – 是否自动调用基类构造函数，默认是。 | class方法用来构造一个类，可派生自任何类。autoCallSuperCtor 设置是否自动调用基类构造函数，若基类和派生类参数一致则自动调用就好。若不一致则设为false，然后在派生类构造函数中手动调用，调用前无法使用派生链中的属性。 |
| typeof | function(obj,classType)   * obj 一个对象或者是类 * classType 一个类 | 判断obj类型是否为classType或classType的派生类，总之就是判断obj是否完整继承classType的行为 |

## Constants

文件说明：

此文件中包含大量常量，框架中所需的所有常量都在这里有定义。

## Global

暂为空，留作他用。

## Anim

文件说明：

动态值anim（也称可变值）是四大对象之一，通过指定起止时间、最小值和最大值，在引擎运行期间动态变化（递增）的整数、浮点数值。

该文件中包含四个基本类，分别是：AnimBase、AnimDouble、AnimInt和AnimIndex, AnimBase是基本动态值类，后面三个类都继承它，有它的一切属性。

Tips:

1. 可变值可用于定时器还可以将可变值赋给属性，然后添加给可绘制对象。
2. 可变值的数目不宜过多（20个以上就要考虑优化），会导致性能下降；另外，尽量避免由于可变值导致的“引擎频繁调用Lua函数”的情况（比如每一帧都调用），这样会导致性能大幅下降；
3. 可变值在引擎内部实现中，每一帧都更新，所以游戏帧数越高可变值越平滑。

### AnimBase

类说明：

AnimBase是最基本的可变值类，它是所有其他可变值类的父类。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| AnimBase.ctor | function(self) | 此为构造函数，new对象时自动调用，利用引擎使其自动生成animID |
| AnimBase.dtor | function(self) | 析构函数，释放animID指定的资源 |
| AnimBase.getCurValue | function(self,defaultValue)  defaultValue 浮点数、整数 | 获取指定animID的当前值，如果失败，则返回defaultValue |
| AnimBase.onEvent | function(self,anim\_type, anim\_id, repeat\_or\_loop\_num)  anim\_type –可变值类型，取值可以为kAnimNormal,kAnimRepeat,kAnimLoop  anim\_id – id  repeat\_or\_loop\_num – 循环次数 | 响应事件 |
| AnimBase.setEvent | function(self,obj,func)  obj,func 同普通回调函数 | 回调接口 |

### AnimDouble

类说明：

生成浮点数类型的动态值类，继承AnimBase，有AnimBase的一切属性。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| AnimDouble.ctor | function(self,animType, startValue, endValue, duration, delay)  animType—可变值类型  startValue, endValue—动态值变化的开始值和结束值。  duration—时间（毫秒级）  delay—延迟时间  delay = {  -1: 从下一帧的开始时间开始计时;  0: 从当前帧的开始时间开始计时;  大于0: 延迟delay毫秒 }; | 此为构造函数，new对象的时候会自动调用，构造一个AnimDouble对象 |
| AnimDouble.dtor | function(self) | 析构函数，释放资源 |

### AnimInt

类说明：

功能与接口都和AnimDouble相同，唯一不同的是产生的可变值为整数。

### AnimIndex

类说明：

功能同上，不同之处在于：可变值本身作为一个数组下标，从【res 整数数组/浮点数数组】中取值。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| AnimIndex.ctor | function(self,animType, startValue, endValue, duration, res, delay)  res —整数或浮点数数组;  其他参数同AnimDouble。 | 此为构造函数，new对象的时候会自动调用，构造一个AnimIndex对象。 |
| AnimIndex.dtor | function(self) | 析构函数 |

## Drawing

文件说明：

Drawing绘制对象是指可以绘制到屏幕的对象，包括点、线、位图、多边形，但绘制对象本身并不包含可绘制的数据，需要引用资源Res的数据，它也不包含绘制的属性，添加属性时需引用属性Prop 对象。

它本文包含6个绘制对象类，为二次开发人员提供了将资源转换成可绘制对象的引擎接口。

Tips：

引擎中的可绘制对象drawing是用树型结构管理的，有如下特性：

1. 对父节点的某些操作会影响子节点，比如：2D变化、可见性、裁剪、颜色；
2. 树状结构定义了渲染次序；
3. 缺省有一个根节点；

### DrawingBase

类说明：

DrawingBase封装了一个基本绘制对象类，拥有绘制对象的一般属性。其它的绘制对象类都继承它的属性，它是所有其它绘制对象类的父类。

对绘制对象设置的所有属性都是继承的，也就是说，若对此节点设置某一属性，则其所有子节点同时继承这一属性。若为enable，disable类型，则优先考虑父节点的disable，也就是说若父节点为disable则不管子节点是否为enable都无用，若父节点为enable，则子节点enable/disable有效。

通过DrawingBase.addProp (self, prop, sequence) 添加的属性，需要自己手动释放，drawing中并不保存这个prop的引用。通过其他方法添加的属性，无需手动释放，直接removeProp就可以了，而且在释放drawing的时候，会完全释放通过这种方法添加到属性。

所有drawing在释放的时候，都会释放以此节点为根的整棵子树。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| DrawingBase.ctor | function(self) | 此为构造函数，无需任何参数，new对象的时候自动调用，为绘制对象自动分配id。 |
| DrawingBase.dtor | function(self) | 析构函数,释放相关信息，包括此节点下的所有子节点 |
| DrawingBase.setPickable | function(self,pickable)  pickable – true / false | 调用该接口设置节点是否可点击，可以继承。若某一节点被设为不可点击，其所有子节点也不可点击；若子节点设为可点击，但父节点不可点击，子节点仍不可点击。 |
| DrawingBase.getPickable | function(self) | 获取某一节点是否可点击的状态 |
| DrawingBase.setEventDoubleClick | function(self, obj, func)  obj, func —同普通回调函数 | 设置双击响应回调接口 |
| DrawingBase.setEventTouch | function(self, obj,func)  obj, func —同普通回调函数 | 触摸事件回调接口 |
| DrawingBase.setPos | function(self,x,y)  x,y —位置坐标 | 设置绘制对象的位置 |
| DrawingBase.getPos | function(self) | 获取当前位置,默认位置为0,0 |
| DrawingBase.setSize | function(self,w,h)  w,h —长宽 | 设置绘制对象的大小 |
| DrawingBase.getSize | function(self) | 返回绘制对象的长、宽，默认大小为0,0 |
| DrawingBase.setColor | function(self, r, g, b)  r,g,b —红绿蓝三个通道的颜色分量，用RGB格式表示，取值为[0,255] | 设置绘制对象的颜色 |
| DrawingBase.getColor | function(self) | 获取当前颜色，默认为255,255,255 |
| DrawingBase.setTransparency | function(self,, val)  val — 透明度，取值为0.0~1.0。 | 调用此接口来对绘制对象的透明度进行设置。若val小于0，则不进行alpha混合，0-1则开启同时设置一个透明度 |
| DrawingBase.getTransparency | function(self) | 获取透明度，默认为1 |
| DrawingBase.setVisible | function(self, visible)  visible – 可见性标志，true可见，false不可见。 | 调用此接口，设置该绘制对象是否可见 |
| DrawingBase.getVisible | function(self) | 获取是否可见，默认可见 |
| DrawingBase.setLevel | function(self, level)  level —绘制次序 | 调用此接口，设置该绘制对象的绘制次序，level数值越大越后被绘制。 |
| DrawingBase.getLevel | function(self) | 获取绘制次序 |
| DrawingBase.setClip | function(self, x,y,w,h)  x,y,w,h —裁剪区的左顶点的坐标(x,y)和裁减区的长宽，如果取值都为-1时则不可裁剪 | 调用该接口设置绘制对象的裁减区 |
| DrawingBase.addProp | function(self, prop, sequence)  prop – 属性对象，可为prop.lua中任一类的实例  sequence —属性添加位置，正整数 | 调用此接口，可为绘制对象添加任意属性，如果sequence已被其他已添加属性暂用则添加失败返回false，否则返回true。 |
| DrawingBase.removeProp | function(self, sequence)  sequence – 意义同上 | 调用此接口，删除指定位置上的属性，如果sequence上无已添加属性则删除失败返回false，否则移除属性返回true。 |
| DrawingBase.removePropByID | function(self, propId)  propId – 与类prop中的m\_propID成员变量相对应 | 调用此接口，根据propId从可绘制对象中移除指定属性，如果该属性未添加在此绘制对象上则删除失败返回false，否则移除属性返回true |
| DrawingBase.addPropColor | function(self,sequence,animType,  duration,delay,  rStart,rEnd,gStart,gEnd,bStart,bEnd)  sequence -- 同addProp  animType --变换类型，可以为kAnimNormal,kAnimRepeat,kAnimLoop  duration – 可变值的周期变化时间  rStart,rEnd,gStart,gEnd,bStart,bEnd—分别表示r,g,b的起始变化值和终止变化值 | 调用此接口，传入给定参数，可以直接添加一个颜色可变属性，其中rStart,rEnd,gStart,gEnd,bStart,bEnd任何一对为空均可，表示此颜色通道不变化。 |
| DrawingBase.addPropTransparency | function(self,sequence,animType,duration,delay,startValue,endValue)  startValue,endValue – 透明度变化 起始值和终止值  其他参数同上 | 调用此接口，传入给定参数，可以直接添加一个透明度可变属性。 |
| DrawingBase.addPropTranslate | function(self,sequence,animType,duration,delay,startX,endX,startY,endY)  startX,endX,startY,endY – x,y方向平移的起始值和终止值 | 调用此接口，传入给定参数，可以直接添加一个可变的平移变换属性，其中startX,endX,startY,endY任意一对均可为空，表示此方向不变化。 |
| DrawingBase.addPropRotate | function(self,sequence,animType,duration,delay,startValue,endValue,center, x, y)  startValue,endValue – 旋转角度起始值和终止值  center – 旋转中心，可以为kNotCenter,kCenterDrawing,kCenterXY  x,y – 若center为kCenterXY,则此参数表示旋转中心坐标  其他参数同上 | 调用此接口，传入给定参数，可以直接添加一个可变的旋转变换属性 |
| DrawingBase.addPropScale | function(self,sequence,animType,duration,delay,startX,endX,startY,endY,center, x, y)  startX,endX,startY,endY – x,y方向上的缩放起始值和终止值  其他参数同上 | 调用此接口，传入给定参数，可以直接添加一个可变的缩放变换属性，其中startX,endX,startY,endY任意一对均可为空，表示此方向不变化。 |
| DrawingBase.addPropTranslateSolid | function(self,sequence, x, y)  x,y – x,y方向上的平移量  sequence – 同上 | 调用此接口，传入给定参数，可以直接添加一个固定的平移变化，其中x,y任意一个均可为nil表示，此方向不变化。 |
| DrawingBase.addPropRotateSolid | function(self,sequence, angle360,center,x,y)  angle360 – 旋转角度  其他参数同addPropRotate | 调用此接口，传入给定参数，可以直接添加一个固定的旋转变化。 |
| DrawingBase.addPropScaleSolid | function(self,sequence,scaleX, scaleY,center,x,y)  scaleX, scaleY – x,y 方向上的缩放变换值  其他参数同addPropScale | 调用此接口，传入给定参数，可以直接添加一个固定的缩放变化，其中scaleX, scaleY任意一个均可为nil表示，此方向不变化。 |
| DrawingBase.setName | function(self,name)  name – 名字 | 设置名字 |
| DrawingBase.getName | function(self) | 获取绘制对象的名字 |
| DrawingBase.addChild | function(self,child)  child –节点，必须是继承或间接继承DrawingBase的类 | 添加子节点 |
| DrawingBase.addToRoot | function(self) | 调用该接口将新生成的节点添加到root节点下。  注意：新创建的节点不在任何节点下，要显示该节点必须添加到某一节点之下。 |
| DrawingBase.removeChild | function(self,child)  child – 添加到DrawingBase中的一个子节点 | 移除子节点 |
| DrawingBase.getParent | function(self) | 返回父节点 |
| DrawingBase.getChildren | function(self) | 获取DrawingBase的所有子节点 |
| DrawingBase.getChildByName | function(self,name)  name –子节点的名字 | 获取指定名称的子节点信息 |
| DrawingBase.convertPointToSurface | function(self,x,y)  x,y –相对于父节点的x,y坐标 | 将子节点相对父节点的偏移坐标(x,y)转换成屏幕坐标(newX,newY)。 |
| DrawingBase.convertSurfacePointToView | function(self,x,y)  x,y –屏幕坐标(x,y) | 将屏幕坐标转换成子节点相对于父节点的偏移坐标 |
| DrawingBase.setScreenRelative | function(self,enable)  enable –可用性标志，enable取 0时不可用，取1时可用 | 调用该接口，设置节点间的相对坐标与屏幕坐标之间是否可以相互转换 |

### DrawingImage

类说明：

DrawingIamge继承自DrawingBase类，所以DrawingBase的成员变量与成员方法，这一切在DrawingIamge中都可以直接使用。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| DrawingImage.ctor | function(self, res,leftWidth,rightWidth,bottomWidth,topWidth)  res -- ResImage的对象  leftWidth,rightWidth,bottomWidth,topWidth -- 9-Grid规范的贴图的4个边缘的宽度，如果其中有一个值为空，则不使用9-Grid贴图。 | 此为构造函数，new对象的时候会自动调用，构造一个DrawingImage对象。 |
| DrawingImage.setResRect | function(self, sequence, x, y, width, height)  sequence --资源存放位置，正整数  x,y --资源的位置坐标  width,height --资源本身的实际长宽，而非显示大小 | 调用此接口设置指定资源的位置与大小。 |
| DrawingImage.setImageIndex | function(self, idx)  idx -- 图片序列的索引，注意图片索引是从0开始的，比如6张图片，其索引为0~5。 | 一个可绘制对象可以设定多个位图，但只能设置一个是当前的，调用此接口用来设置当前要绘制的位图的索引。 |
| DrawingImage.addImage | function(self, res, sequence)  res --ResImage对象  sequence—位图资源存放位置，正整数 | 调用该接口往指定的序列sequence中添加位图资源。 |
| DrawingImage.removeImage | function(self, sequence)  sequence同上。 | 移除指定序列中的位图资源。 |
| DrawingImage.removeAllImage | function(self) | 移除所有的位图资源。 |
| DrawingImage.dtor | function(self) | 析构函数，用来释放该绘制对象所占的资源。 |

### DrawingEmpty

类说明：

DrawingEmpty封装的是一个空的绘制对象类，它继承DrawingBase，具有DrawingBase的所有属性，能直接使用DrawingBase的成员变量与成员方法。

## Prop

文件说明：

计算机内的属性Prop是用来描述实体的性质或特征的，通常具有数据类型、域、默认值三种性质。而引擎内的属性Prop对象是用于对控件特性的描述，包括了颜色、点大小、线宽、透明度、2D变化（平移、旋转、缩放）、索引、起止点等等，属性的值可以是固定值，也可以是可变值anim。

Tips：

引擎内部是用attribute命名的，引擎提供给lua的接口是用prop命名的。

### PropBase

类说明：

PropBase是最基本的属性类，它是所有其它属性类的父类。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropBase.ctor | function(self) | 此为构造函数，new对象时自动调用，为 属性对象自动分配id。 |
| PropBase.dtor | function(self) | 析构函数，用于释放指定id所占的内存资源。 |

### PropTranslate

类说明：

PropTranslate是对平移属性的一个简单封装，只要传入必要的参数就可创建一个2D平移属性。它继承PropBase，具有PropBase的所有属性。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropTranslate.ctor | function(self, animX, animY)  animX, animY—可变值anim对象，通过读取animX, animY中的值来指定x、 y方向上平移的距离。 | 此为构造函数，new对象时自动调用，构造一个2d平移属性PropTranslate对象。  animX, animY任何一个值为空则表示目标在该方向上不平移。 |

### PropTranslateSolid

类说明：

PropTranslateSolid是固定平移属性的封装，继承PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropTranslateSolid.ctor | function(self, x, y)  x,y—double/int类型的值，用来指定目标在x、y方向上平移的距离。 | 此为构造函数，new对象时自动调用，构造一个固定平移属性PropTranslateSolid的对象。  x, y任何一个值为空则表示目标在该方向上不平移。 |
| PropTranslateSolid.set | function(self, x, y)  x,y—同上。 | 调用此接口，设置目标在x方向、y方向上平移的距离。 |

### PropRotate

类说明：

PropRotate旋转属性类，继承PropBase，具有PropBase的所有属性。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropRotate.ctor | function(self, anim, center, x, y)  anim—anim对象，此处用作定时器，指定旋转的时间。  center – 旋转中心，可以为kNotCenter,kCenterDrawing,kCenterXY。center取kNotCenter时，目标是将其左顶点作为旋转中心；取kCenterDrawing时，则是以绘制对象的中心作为旋转中心；取kCenterXY时则需要传入 (x,y)作为旋转中心。  x,y – 若center为kCenterXY,则此参数表示旋转中心坐标。 | 此为构造函数，new对象时自动调用，构造一个旋转属性PropRotate的对象。 |

### PropRotateSolid

类说明：

PropRotateSolid固定旋转类，继承类PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropRotateSolid.ctor | function(self, angle360,center,x,y)  angle360—固定旋转的角度  其他参数同PropRotate。 | 此为构造函数，new对象时自动调用，构造一个固定旋转属性PropRotateSolid的对象。注意：每构造一次固定旋转属性、固定平移属性等目标都只旋转或平移一次。 |
| PropRotateSolid.set | function(self, angle360)  angle360—固定旋转的角度 | 调用此接口设置固定旋转的角度。 |

### PropScale

类说明：

PropScale缩放属性类，继承类PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropScale.ctor | function(self, animX, animY, center,x,y)  animX, animY—可变值anim对象，用来指定x,y方向上的缩放变换值。  center—缩放中心，可以为kNotCenter,kCenterDrawing,kCenterXY。  x,y—若center为kCenterXY,则此参数表示缩放中心坐标。 | 此为构造函数，new对象时自动调用，构造一个缩放属性PropScale对象。  animX, animY任何一个值为空则表示目标在该方向上不缩放。 |

### PropScaleSolid

类说明：

PropScaleSolid固定缩放，继承类PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropScaleSolid.ctor | function(self, scaleX, scaleY,center,x,y)  scaleX, scaleY—x,y 方向上的缩放变换值  其他参数同PropScale。 | 此为构造函数，new对象时自动调用，构造一个固定缩放属性PropScaleSolid对象。  scaleX, scaleY任何一个值为空则表示目标在该方向上不缩放。 |
| PropScaleSolid.set | function(self, scaleX, scaleY)  参数同上 | 调用该接口，设置目标在x,y方向上的固定缩放的值。 |

### PropColor

类说明：

PropColor颜色可变属性类，继承PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropColor.ctor | function(self, animR, animG, animB)  animR, animG, animB—anim对象，用来指定R G B各颜色分量的当前值。可变值anim如果为int类型，则需要0~255方式描述；若为double类型或index类型，则需要0.0~1.0方式描述； | 构造函数，new对象时自动调用，构造一个颜色可变属性PropColor对象。  animR, animG, animB任何一个为空，则表示此通道的颜色不发生变化。 |

### PropPointSize

类说明：

PropPointSize是点大小可变的属性类，继承PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropPointSize.ctor | function(self, anim)  anim—anim对象，读取anim当前值指定点的当前大小，通过anim的变化实现点的大小的变化。 | 构造函数，new对象的时候自动调用，构造一个PropPointSize的对象。 |

### PropLineWidth

类说明：

PropLineWidth是线宽可变的属性类，继承类PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropLineWidth.ctor | function(self, anim)  参数同PropPointSize。 | 此为构造函数，new对象时自动调用，构造一个线宽属性PropLineWidth对象。 |

### PropTransparency

类说明：

PropTransparency封装的是透明度变化属性类，继承PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropTransparency.ctor | function(self, anim)  anim —anim对象，通过读取anim的值指定目标透明度的值，通过anim的变化实现目标透明度的变化。可变值anim的范围建议为0.0~1.0。 | 此为构造函数，new对象时会自动调用，构造一个透明度变化属性类PropTransparency的对象。 |

### PropImageIndex

类说明：

PropImageIndex索引属性，用于可绘制对象drawing来实现精灵（Sprite），继承PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropImageIndex.ctor | function(self, anim)  anim—用于指定绘制对象的当前索引。 | 此为构造函数，new对象时会自动调用，构造一个索引属性类PropImageIndex的对象。 |

### PropClip

类说明：

PropClip裁剪区，继承PropBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| PropClip.ctor | function(self, animX,animY,animW,animH)  animX,animY —裁剪区的位置  animW,animH —裁减区的宽高 | 此为构造函数，new对象时会自动调用，构造一个PropClip的对象。其中animX,animY,animW,animH任何一个值为空，裁减区就不可用。 |

## Res

文件说明：

资源Res包括矢量和位图两大类；其中矢量是整数、浮点数缓冲（内部是数组）；位图是像素缓冲（内部是OpenGL贴图）。

Tips:

OpenGL中的资源都是显存的，res\_free\_xxx与 res\_reload\_xx等接口只是释放、加载资源所占用的内存，并不销毁或重建资源本身，要销毁资源必须调用res\_delete和res\_delete\_group接口。

### ResBase

类说明：

ResBase是最基本的资源类，它是所有其它资源类的父类。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| ResBase.ctor | function(self) | 此为构造函数，new对象时自动调用，为 资源对象自动分配id。 |
| ResBase.dtor | function(self) | 析构函数，用来释放资源所占的内存。 |

### ResImage

类说明：

ResImage是资源的位图类，继承ResBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| ResImage.ctor | function(self, file, imageFormat, filter)  file —文件名  imageFormat —图片加载格式，可为kRGBA8888,kRGBA4444,kRGBA5551, kRGB565  filter —纹理贴图过滤方式，可为kFilterNearest,kFilterLinear | 此为构造函数，new对象的时候会自动调用，构造一个ResImage对象。 |
| ResImage.releaseImage | function(self) | 调用该接口释放位图资源所占的内存，注意该接口并不销毁资源对象本身 |
| ResImage.reloadImage | function(self) | 调用此接口重新加载位图资源。 |
| ResImage.setFile | function(self, file, imageFormat, filter)  参数同构造函数。 | 调用此接口，可改变位图资源。 |
| ResImage.dtor | function(self) | 析构函数，调用此接口销毁资源。 |

### ResText

类说明：

ResText文本资源类，继承ResBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| ResText.ctor | function(self, str, width, height, align, fontName, fontSize, r, g, b, multiLines)  str—要创建的文本内容  width,height – 宽高  align 对其方式，在宽高足够时生效  fontName – 字体名  fontSize – 字体大小  r,g,b – 字符颜色  multiLines为1是多行，为0是单行 | 构造函数，创建一个文本资源对象。 |
| ResText.setText | function(self,str,width,height)  参数同构造函数。 | 重新设置文本内容和宽高 |
| ResText.dtor | function(self) | 析构函数，销毁资源。 |

### ResDoubleArray

类说明：

ResDoubleArray自定义浮点数组，继承ResBase。

接口说明：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数原型及参数说明 | 接口说明 |
| ResDoubleArray.ctor | function(self, nums)  nums – 是一个lua 的table，从1开始无空洞的数字索引数组。 | 此为构造函数，new对象时自动调用，构造一个ResDoubleArray对象。 |
| ResDoubleArray.setData | function(self, nums)  参数同上。 | 调用此接口重置浮点数组中的值。 |
| ResDoubleArray.ctor | function(self) | 析构函数，销毁资源。 |

### ResIntArray

类说明：

ResIntArray自定义整数数组，它的功能、接口等都和ResDoubleArray一样，只是生成的是整数类型的数组。