## **Bitmap**

**save\_png(String path)**

**保存位图数据为png并写入磁盘指定位置**

**process\_pixel() { |PixelArray pixels| block }**

**在内存中处理位图中的像素数据，不要在提供的块外使用PixelArray的方法**

**块内的代码执行完毕后就会将内存数据刷新到显存中**

## **PixelArray - 用于Bitmap内部处理像素**

**[Integer index] -> Integer**

**获取指定位置的像素数据，返回一个点的像素数据整数，需要自己解析为单个通道值**

**本引擎的位图格式统一为ABGR8888**

**[Integer index, Integer pixel]=**

**向指定位置的像素设置新的像素，像素格式参考上一个函数**

**size -> Integer**

**获取位图的像素大小，实际为 长 \* 宽 \* 4**

**save\_data(String buffer)**

**将像素数据保存到String中，注意在保存前需要提前申请内存：**

**Str = ‘\0’\* size**

**load\_data(String buffer)**

**将数据从String中读入到位图对象中，注意保证String的大小≥size**

## **Viewport**

**snap\_to\_bitmap(Bitmap target)**

**将视口中的各种活动块渲染到位图中，可无视视口的可视属性直接绘制，**

**绘制的对象会保留视口的位置偏移，同时会清空位图中原先的数据而不是覆盖渲染**

## **Input**

**press\_key?(Integer scancode)**

**判断是否按下某个按键，相比原版支持全键盘，**

**有关按键scancode请参考sdl的头文件：**

[**https://github.com/libsdl-org/SDL/blob/main/include/SDL3/SDL\_scancode.h**](https://github.com/libsdl-org/SDL/blob/main/include/SDL3/SDL_scancode.h)

**trigger\_key?(Integer scancode)**

**判断是否触发一个按键**

**repeat\_key?(Integer scancode)**

**判断是否按住某个按键**

**recent\_pressed -> Array**

**recent\_triggered -> Array**

**recent\_repeated -> Array**

**返回最近按下的按键**

**get\_key\_name(Integer scancode) -> String**

**通过按键code获取按键名称**

**get\_keys\_from\_flag(String flags) -> Array**

**set\_keys\_from\_flag(String flags, Array keys)**

**设置按键绑定的位置，可实现F1的相同功能**

## **RGU – 引擎内置扩展模块**

**CONTENTVERSION -> Integer**

**常量：引擎识别的RGSS版本（1-3）**

**SDLVERSION -> Integer**

**常量：引擎使用的SDL版本**

**get\_locale -> Hash**

**获取本机的地区及语言**

**Hash: { “country” : “US”,**

**“language” : “en”, }**

**open\_url(String url)**

**使用本机默认浏览器打开一个网址**

## **Mouse – 鼠标扩展模块**

**update**

**更新鼠标状态，每帧调用**

**x y**

**获取鼠标的相对位置，如果窗口有缩放，则返回计算比例后相对于分辨率的坐标**

**set\_pos(Integer x, Integer y)**

**设置鼠标位置，相对分辨率设置**

**down?(Integer id)**

**判断指定鼠标按键是否刚刚按下**

**up?(Integer id)**

**判断指定鼠标按键是否刚刚弹起**

**double\_click?(Integer id)**

**判断指定鼠标按键是否点击次数=2**

**press?(Integer id)**

**判断指定鼠标按键是否被按下**

**scroll\_x scroll\_y**

**获取鼠标滚轮的累计滚动坐标**

**cursor([Bitmap cursor, Integer hot\_x, Integer hot\_y])**

**设置鼠标图像和热点位置**

**visible**

**visible=**

**设置鼠标在当前窗口的可见状态**

**LEFT MIDDLE RIGHT**

**X1 X2**

**模块内置的按钮ID常量**

**Geometry – 批量绘制三角形的类**

**new([Viewport viewport])**

**创建Geometry，初始三角形容量为64**

**其中每个像素点的合成方式为：**

**纹理的颜色 \* (1 – 顶点颜色的不透明度) + 顶点颜色 \* 顶点颜色的不透明度**

**像素点的不透明度继承纹理不透明度**

**如果没有指定纹理，则纹理颜色默认为(0, 0, 0,1) -> RGBA**

**viewport viewport=**

**z z=**

**visible visible=**

**dispose disposed?**

**与原版相同功能**

**capacity -> Integer**

**获取当前类的最大三角形容量**

**resize(Integer size)**

**重新扩展最大三角形容量，之前的数据不会丢弃**

**set\_position(Numeric index, Integer x, Integer y, Numeric z, Numeric w)**

**为编号为index的顶点设置属性**

**总顶点容量为三角形容量x3**

**坐标中的x, y是屏幕坐标，会自动变换为RGSS的坐标系**

**坐标中的z, w是glsl中的坐标，一般z为0，w为1**

**set\_texcoord(Numeric index, Integer tex\_x, Integer tex\_y)**

**设置顶点的纹理坐标**

**set\_color(Numeric index, Color color)**

**设置顶点的颜色数据**

**bitmap bitmap=**

**blend\_type blend\_type=**

**与其他绘制类相同用法**