

PainterEngine API manual

[PainterEngine API manual 1](#_Toc72669296)

[Core 9](#_Toc72669297)

[预编译 9](#_Toc72669298)

[计算数组个数 9](#_Toc72669299)

[日志/调试信息 10](#_Toc72669300)

[输出日志信息 10](#_Toc72669301)

[错误信息 10](#_Toc72669302)

[断言 10](#_Toc72669303)

[取得日志文本 10](#_Toc72669304)

[内存 11](#_Toc72669305)

[内存拷贝 11](#_Toc72669306)

[内存设值 11](#_Toc72669307)

[按4字节对齐设置内存值 11](#_Toc72669308)

[内存比对 11](#_Toc72669309)

[内存池 12](#_Toc72669310)

[创建内存池 12](#_Toc72669311)

[返回一块内存指针所指向内存的内存大小 12](#_Toc72669312)

[从内存池分配一块内存 12](#_Toc72669313)

[从内存池释放一块内存 13](#_Toc72669314)

[重置内存池 13](#_Toc72669315)

[设置内存池异常回调 14](#_Toc72669316)

[字符串操作 14](#_Toc72669317)

[转换 19](#_Toc72669318)

[字节类型转十六进制字符串 19](#_Toc72669319)

[十六进制字符串转字节类型 19](#_Toc72669320)

[十六进制字符转数字 19](#_Toc72669321)

[字符转数字 20](#_Toc72669322)

[字符串转IPv4地址 20](#_Toc72669323)

[IPv4地址转字符串 20](#_Toc72669324)

[数字转字符串 20](#_Toc72669325)

[弧度角度转换 21](#_Toc72669326)

[字节序 22](#_Toc72669327)

[主机字节序转网络字节序 22](#_Toc72669328)

[网络字节序转主机字节序 22](#_Toc72669329)

[数学 22](#_Toc72669330)

[素数模乘同余发生器 22](#_Toc72669331)

[高斯随机数发生器 23](#_Toc72669332)

[对数函数 23](#_Toc72669333)

[三角函数 24](#_Toc72669334)

[平方根 24](#_Toc72669335)

[方差 25](#_Toc72669336)

[协方差 25](#_Toc72669337)

[求绝对值 25](#_Toc72669338)

[tanh 25](#_Toc72669339)

[sigmoid 26](#_Toc72669340)

[ReLU 26](#_Toc72669341)

[四舍五入 26](#_Toc72669342)

[幂 26](#_Toc72669343)

[复数 27](#_Toc72669344)

[矩阵 28](#_Toc72669345)

[点/向量 30](#_Toc72669346)

[交叉碰撞 33](#_Toc72669347)

[快速排序(由小到大) 33](#_Toc72669348)

[快速排序(由大到小) 34](#_Toc72669349)

[矩形 34](#_Toc72669350)

[CRC校验（CRC32） 35](#_Toc72669351)

[CRC校验（CRC16） 35](#_Toc72669352)

[累加和校验（Sum32） 35](#_Toc72669353)

[第一类修正贝塞尔函数 36](#_Toc72669354)

[通讯链接器(IO虚拟层) 36](#_Toc72669355)

[初始化链接器 36](#_Toc72669356)

[链接器读 37](#_Toc72669357)

[链接器写 37](#_Toc72669358)

[数据结构 37](#_Toc72669359)

[取得结构体偏移量 37](#_Toc72669360)

[字母 37](#_Toc72669361)

[链表 38](#_Toc72669362)

[容器 41](#_Toc72669363)

[字符串映射表 43](#_Toc72669364)

[自适应内存 45](#_Toc72669365)

[内存自适应字符串 48](#_Toc72669366)

[几何绘制 52](#_Toc72669367)

[线 52](#_Toc72669368)

[边框 53](#_Toc72669369)

[矩形 53](#_Toc72669370)

[实心圆 53](#_Toc72669371)

[圆 54](#_Toc72669372)

[圆笔 54](#_Toc72669373)

[环 54](#_Toc72669374)

[扇形 55](#_Toc72669375)

[圆角矩形 55](#_Toc72669376)

[实心圆角矩形 56](#_Toc72669377)

[描边路径 56](#_Toc72669378)

[三角形 56](#_Toc72669379)

[箭头 57](#_Toc72669380)

[绘制贝塞尔曲线 57](#_Toc72669381)

[图形图像 58](#_Toc72669382)

[渲染表面 58](#_Toc72669383)

[纹理 61](#_Toc72669384)

[图像算子 70](#_Toc72669385)

[轮廓 72](#_Toc72669386)

[位图 74](#_Toc72669387)

[TRaw纹理格式 76](#_Toc72669388)

[Delaunary三角划分 77](#_Toc72669389)

[带限制区域的Delaunary三角建立 77](#_Toc72669390)

[纹理特效 78](#_Toc72669391)

[字体 78](#_Toc72669392)

[绘制字体 78](#_Toc72669393)

[字模初始化 79](#_Toc72669394)

[加载字模 79](#_Toc72669395)

[释放字模库 79](#_Toc72669396)

[绘制单个字 80](#_Toc72669397)

[计算绘制文本的宽高 80](#_Toc72669398)

[信号处理 80](#_Toc72669399)

[离散傅里叶变换（DFT） 80](#_Toc72669400)

[离散余弦变换（DCT） 81](#_Toc72669401)

[快速傅里叶变换（FFT） 81](#_Toc72669402)

[强制共轭对称 82](#_Toc72669403)

[上采样(插值, UpSampled) 82](#_Toc72669404)

[下采样(抽取, SubSampled, DownSampled) 83](#_Toc72669405)

[tukey窗 83](#_Toc72669406)

[triangular窗 83](#_Toc72669407)

[blackMan窗 83](#_Toc72669408)

[hamming窗 84](#_Toc72669409)

[hanning窗 84](#_Toc72669410)

[kaiser窗 84](#_Toc72669411)

[应用窗函数 84](#_Toc72669412)

[计算系统幅频响应和相频响应 85](#_Toc72669413)

[计算级联型系统幅频响应和相频响应 85](#_Toc72669414)

[倒谱 85](#_Toc72669415)

[人声基音频率估算 86](#_Toc72669416)

[估算瞬时频率 86](#_Toc72669417)

[预加重 86](#_Toc72669418)

[卡尔曼滤波 87](#_Toc72669419)

[MFCC 87](#_Toc72669420)

[WAV文件格式 88](#_Toc72669421)

[混音器 89](#_Toc72669422)

[调音台 92](#_Toc72669423)

[反向传播神经网络 95](#_Toc72669424)

[初始化神经网络框架 95](#_Toc72669425)

[添加网络层 95](#_Toc72669426)

[训练网络 96](#_Toc72669427)

[前向传播 96](#_Toc72669428)

[取得输出结果 97](#_Toc72669429)

[重置神经网络 97](#_Toc72669430)

[释放神经网络框架 97](#_Toc72669431)

[导出神经网络框架 97](#_Toc72669432)

[导入神经网络框架 97](#_Toc72669433)

[物理 98](#_Toc72669434)

[四叉树AABB碰撞检测 98](#_Toc72669435)

[密码学 99](#_Toc72669436)

[AES 99](#_Toc72669437)

[curve25519 100](#_Toc72669438)

[SHA256 101](#_Toc72669439)

[编码及数据压缩 101](#_Toc72669440)

[BASE64 101](#_Toc72669441)

[ARLE 102](#_Toc72669442)

[Huffman 103](#_Toc72669443)

[3D渲染管线 103](#_Toc72669444)

[Kernel 109](#_Toc72669445)

[词法分析机 109](#_Toc72669446)

[初始化词法分析机 109](#_Toc72669447)

[注册注释块 109](#_Toc72669448)

[注册包含块 109](#_Toc72669449)

[注册空白符 110](#_Toc72669450)

[注册分隔符 110](#_Toc72669451)

[取得分隔符ID 110](#_Toc72669452)

[取得字符串包含块类型 110](#_Toc72669453)

[释放词法分析机资源 111](#_Toc72669454)

[加载分析文本 111](#_Toc72669455)

[读取文本 111](#_Toc72669456)

[取得特殊符号 111](#_Toc72669457)

[取得当前词 112](#_Toc72669458)

[取得包含文本 112](#_Toc72669459)

[设置词大小写敏感 112](#_Toc72669460)

[取得词法分析机当前状态 112](#_Toc72669461)

[设置词法分析机当前状态 113](#_Toc72669462)

[判断当前词是否是合法数字 113](#_Toc72669463)

[读取下一个字母 113](#_Toc72669464)

[读取下一个词 113](#_Toc72669465)

[取得当前词性 114](#_Toc72669466)

[PainterScript语法规则 114](#_Toc72669467)

[脚本语法 114](#_Toc72669468)

[PainterScript 简化编译器 126](#_Toc72669469)

[初始化编译器 126](#_Toc72669470)

[添加源代码 126](#_Toc72669471)

[编译源代码 126](#_Toc72669472)

[释放编译器 127](#_Toc72669473)

[PainterScript Asm汇编语法 127](#_Toc72669474)

[标志 127](#_Toc72669475)

[汇编指令集 127](#_Toc72669476)

[表达式 138](#_Toc72669477)

[PainterScript Asm汇编器 138](#_Toc72669478)

[编译汇编代码 138](#_Toc72669479)

[PainterVM虚拟机 139](#_Toc72669480)

[vm实例 139](#_Toc72669481)

[host函数定义 139](#_Toc72669482)

[执行虚拟机函数 140](#_Toc72669483)

[运行虚拟机 141](#_Toc72669484)

[线程上下文切换 142](#_Toc72669485)

[终止线程 142](#_Toc72669486)

[挂起线程 142](#_Toc72669487)

[恢复线程 143](#_Toc72669488)

[注册host函数 143](#_Toc72669489)

[释放虚拟机 143](#_Toc72669490)

[类型访问 143](#_Toc72669491)

[命令调试系统 145](#_Toc72669492)

[Obj静态3D模型加载器 145](#_Toc72669493)

[加载器初始化 145](#_Toc72669494)

[解析Object数据 146](#_Toc72669495)

[数据转换到RenderList 146](#_Toc72669496)

[释放加载器 146](#_Toc72669497)

[2dx动画描述语言 146](#_Toc72669498)

[指令 147](#_Toc72669499)

[编译 148](#_Toc72669500)

[2dx动画 148](#_Toc72669501)

[2dx library 148](#_Toc72669502)

[2dx animation 149](#_Toc72669503)

[PainterEngine LiveFramework 153](#_Toc72669504)

[导入LiveFramework 153](#_Toc72669505)

[LiveFramework动画播放 154](#_Toc72669506)

[LiveFramework暂停播放 154](#_Toc72669507)

[重置LiveFramework 154](#_Toc72669508)

[停止LiveFramework 155](#_Toc72669509)

[渲染LiveFramework 155](#_Toc72669510)

[PainterEngine Live 155](#_Toc72669511)

[创建Live 155](#_Toc72669512)

[释放Live 156](#_Toc72669513)

[Live动画播放 156](#_Toc72669514)

[Live暂停播放 156](#_Toc72669515)

[重置live 156](#_Toc72669516)

[停止live 157](#_Toc72669517)

[渲染live 157](#_Toc72669518)

[粒子系统 157](#_Toc72669519)

[粒子发射器参数 157](#_Toc72669520)

[以默认值初始化一个发射器参数 158](#_Toc72669521)

[创建粒子系统 159](#_Toc72669522)

[渲染粒子发射器 159](#_Toc72669523)

[释放粒子发射器 159](#_Toc72669524)

[设置粒子发射器的位置 159](#_Toc72669525)

[设置粒子发射器喷射方向 160](#_Toc72669526)

[MQTT协议 160](#_Toc72669527)

[初始化协议 160](#_Toc72669528)

[MQTT连接 160](#_Toc72669529)

[发布数据 161](#_Toc72669530)

[订阅数据 161](#_Toc72669531)

[取消订阅数据 162](#_Toc72669532)

[MQTT Ping 162](#_Toc72669533)

[监听 162](#_Toc72669534)

[释放 163](#_Toc72669535)

[资源管理器 163](#_Toc72669536)

[初始化 163](#_Toc72669537)

[加载一个资源 163](#_Toc72669538)

[添加一个贴图资源 164](#_Toc72669539)

[读取资源 164](#_Toc72669540)

[读取纹理资源 164](#_Toc72669541)

[读取轮廓资源 165](#_Toc72669542)

[读取动画资源 165](#_Toc72669543)

[读取脚本资源 165](#_Toc72669544)

[读取声音资源 165](#_Toc72669545)

[读取数据资源 166](#_Toc72669546)

[删除资源 166](#_Toc72669547)

[释放资源库 166](#_Toc72669548)

[帧同步协议 166](#_Toc72669549)

[同步服务端 167](#_Toc72669550)

[帧客户端 169](#_Toc72669551)

[JSON 170](#_Toc72669552)

[初始化 171](#_Toc72669553)

[读取JSON Object的子数据 171](#_Toc72669554)

[读取JSON数据 171](#_Toc72669555)

[读取JSON Array数据 172](#_Toc72669556)

[解析JSON数据 172](#_Toc72669557)

[建立JSON数据类型 172](#_Toc72669558)

[将JSON结构转换为文本 172](#_Toc72669559)

[删除JSON结构中一个数据 173](#_Toc72669560)

[释放JSON 173](#_Toc72669561)

[JSON数据建立 173](#_Toc72669562)

[PainterEngine 基本对象 178](#_Toc72669563)

[使用PainterEngine绘制对象 178](#_Toc72669564)

[PainterEngine对象机制 178](#_Toc72669565)

[创建对象 184](#_Toc72669566)

[设置对象的usercode 185](#_Toc72669567)

[删除对象 185](#_Toc72669568)

[对象属性设置 185](#_Toc72669569)

[PainterEngine更新/绘制流水线 186](#_Toc72669570)

[对象事件机制 187](#_Toc72669571)

[事件类型 187](#_Toc72669572)

[响应处理函数 188](#_Toc72669573)

[注册对象事件 188](#_Toc72669574)

[派分事件 189](#_Toc72669575)

[构造事件 189](#_Toc72669576)

[构造字符串事件 189](#_Toc72669577)

[指针事件偏移 190](#_Toc72669578)

[取得指针事件的X坐标 190](#_Toc72669579)

[取得指针事件的Y坐标 190](#_Toc72669580)

[取得指针事件的Z坐标 190](#_Toc72669581)

[设置指针事件的X坐标 191](#_Toc72669582)

[设置指针事件的Y坐标 191](#_Toc72669583)

[设置指针事件的X坐标 191](#_Toc72669584)

[取得键盘事件的输入码 191](#_Toc72669585)

[取得输入法事件的输入字符串 192](#_Toc72669586)

[PainterEngine UI对象 192](#_Toc72669587)

[静态文本框 192](#_Toc72669588)

[进度条 194](#_Toc72669589)

[图片框 196](#_Toc72669590)

[滑动框 197](#_Toc72669591)

[按钮 199](#_Toc72669592)

[文本编辑框 203](#_Toc72669593)

[列表框 207](#_Toc72669594)

[虚拟键盘 210](#_Toc72669595)

[勾选框 213](#_Toc72669596)

[菜单 216](#_Toc72669597)

[下拉框 217](#_Toc72669598)

[单选框 220](#_Toc72669599)

[文件浏览器 223](#_Toc72669600)

[子窗口 226](#_Toc72669601)

[消息对话框 230](#_Toc72669602)

[PainterEngine UI设计框架 232](#_Toc72669603)

[基本数据类型 232](#_Toc72669604)

[UI类型及对应属性 233](#_Toc72669605)

[label静态文本框 233](#_Toc72669606)

[processbar进度条 233](#_Toc72669607)

[image图像框 234](#_Toc72669608)

[sliderbar 滑动框 234](#_Toc72669609)

[pushbutton/cursorbutton按钮 234](#_Toc72669610)

[edit文本编辑框 234](#_Toc72669611)

[scrollarea滚动区域 235](#_Toc72669612)

[autotext自动换行静态文本框 235](#_Toc72669613)

[virtualkeyboard虚拟键盘 235](#_Toc72669614)

[virtualnumberkeyboard虚拟数字键盘 235](#_Toc72669615)

[checkbox 选择框 236](#_Toc72669616)

[radiobox单选框 236](#_Toc72669617)

[selectbar下拉框 236](#_Toc72669618)

[Widget 子窗体 237](#_Toc72669619)

[游戏世界框架 237](#_Toc72669620)

[游戏世界初始化 237](#_Toc72669621)

[取得游戏世界Object的个数 237](#_Toc72669622)

[删除Object 238](#_Toc72669623)

[在世界查找Object 238](#_Toc72669624)

[添加Object 238](#_Toc72669625)

[更新游戏世界 239](#_Toc72669626)

[渲染游戏世界 239](#_Toc72669627)

[设置摄像机 239](#_Toc72669628)

[设置游戏对象的碰撞类型 239](#_Toc72669629)

[设置游戏世界辅助线间隔 240](#_Toc72669630)

[设置游戏世界辅助线 240](#_Toc72669631)

[设置游戏世界辅助线的颜色 240](#_Toc72669632)

[世界坐标转换为屏幕坐标 241](#_Toc72669633)

[释放游戏世界 241](#_Toc72669634)

## Core

### 预编译

#### 计算数组个数

|  |  |
| --- | --- |
| 函数名 | PX\_COUNTOF(x) |
| 说明 | **计算一个数组的个数** |
| 参数 | fmt 日志信息 |
| 返回值 |  |

### 日志/调试信息

#### 输出日志信息

|  |  |
| --- | --- |
| 函数名 | void PX\_LOG(char fmt[]); |
| 说明 | **输出一段日志信息** |
| 参数 | fmt 日志信息 |
| 返回值 |  |

#### 错误信息

|  |  |
| --- | --- |
| 函数名 | void PX\_ERROR(char fmt[]); |
| 说明 | **输出一段错误日志信息** |
| 参数 | fmt 日志信息 |
| 返回值 |  |

#### 断言

|  |  |
| --- | --- |
| 函数名 | void PX\_ASSERT(void); |
| 说明 | **断言函数,debug模式下执行该函数将触发一个异常中断** |
| 参数 | fmt 日志信息 |
| 返回值 |  |

#### 取得日志文本

|  |  |
| --- | --- |
| 函数名 | char \*PX\_GETLOG(void); |
| 说明 | **取得日志文本** |
| 参数 |  |
| 返回值 | 日志文本指针 |

### 内存

#### 内存拷贝

|  |  |
| --- | --- |
| 函数名 | void px\_memcpy(void \*dst,void \*src,px\_uint size); |
| 说明 | **内存拷贝** |
| 参数 | dst:指向目标内存  src:指向源内存  size:需要拷贝的内存大小 |

#### 内存设值

|  |  |
| --- | --- |
| 函数名 | void px\_memset(void \*dst,px\_byte byte,px\_uint size) |
| 说明 | **设置内存值** |
| 参数 | Dst 目标内存指针  Byte 值  Size 设置内存大小 |

#### 按4字节对齐设置内存值

|  |  |
| --- | --- |
| 函数名 | void px\_memdwordset(void \*dst,px\_dword dw,px\_uint count) |
| 说明 | **按4字节对齐设置内存值** |
| 参数 | Dst 目标内存指针  dw 值  count 需要设置的四字节个数 |

#### 内存比对

|  |  |
| --- | --- |
| 函数名 | px\_bool px\_memequ(void \*dst,void \*src,px\_uint size) |
| 说明 | **比较目标内存是否相同** |
| 参数 | Dst 目标内存指针  src 目标内存指针  Size 比较大小 |
| 返回值 | 如果相同返回PX\_TRUE,否者返回PX\_FALSE |

### 内存池

#### 创建内存池

//Create a memory pool & return a MemoryPool structure

//MemoryAddr :Start address of memory

//MemorySize :Size of memory pool

|  |  |
| --- | --- |
| 函数名 | MemoryPool MP\_Create (void \*MemoryAddr,unsigned int MemorySize); |
| 说明 | **创建一个内存池** |
| 参数 | MemoryAddr :内存池的开始地址  MemorySize :需要创建的内存池的大小  返回一个MemoryPool结构体 |
| 返回值 | 内存池结构 |

#### 返回一块内存指针所指向内存的内存大小

//Get memory size of Ptr

//Pool: Pool MemoryPool structure pointer

//Ptr: memory pointer

//Return - if succeeded return the size of Ptr,else return zero

|  |  |
| --- | --- |
| 函数名 | px\_uint MP\_Size(px\_memorypool \*Pool,px\_void \*Ptr); |
| 说明 | **返回一块内存指针所指向内存的内存大小** |
| 参数 | Pool 内存池  Ptr 指针 |
| 返回值 | 该指针所指向内存块大小 |

#### 从内存池分配一块内存

//Alloc a memory from memory pool

//Pool: Pool MemoryPool structure pointer

//Size: Size of alloc

//Return - if succeeded return the begin address of memories

// if faith return null

|  |  |
| --- | --- |
| 函数名 | **void \*MP\_Malloc (MemoryPool \*Pool,unsigned int Size);** |
| 说明 | **从内存池分配一块内存**  **该内存将会优先从已回收内存节点中分配,如果回收节点无法分配再从空余空间分配** |
| 参数 | Pool：指向MemoryPool的结构体，这个结构体使用MP\_Create进行创建  Size：需要分配的大小  如果成功返回这个内存的指针，如果失败返回null/0 |
| 返回值 | 分配的内存指针,如果分配失败将会返回PX\_NULL |

#### 从内存池释放一块内存

//Free the memory from memory pool

//Pool: Pool MemoryPool structure pointer

//pAddress: Pointer memory need to be free

Pool：指向MemoryPool的结构体，这个结构体使用MP\_Create进行创建

pAddress:内存首地址指针

**void MP\_Free (MemoryPool \*Pool,void \*pAddress);**

void MP\_Release (MemoryPool \*Pool);

|  |  |
| --- | --- |
| 函数名 | **void MP\_Free (MemoryPool \*Pool,void \*pAddress);** |
| 说明 | **释放一个内存节点** |
| 参数 | Pool：指向MemoryPool的结构体，这个结构体使用MP\_Create进行创建  pAddress:内存首地址指针 |
| 返回值 | - |

#### 重置内存池

void MP\_Reset (MemoryPool \*Pool);

|  |  |
| --- | --- |
| 函数名 | **void MP\_Reset (MemoryPool \*Pool);** |
| 说明 | **重置内存池,** **重置内存池将该内存池所有内存分配节点快速释放,对于某类归类或计算算法的加速尤为有用** |
| 参数 | Pool：指向MemoryPool的结构体，这个结构体使用MP\_Create进行创建 |
| 返回值 | - |

#### 设置内存池异常回调

|  |  |
| --- | --- |
| 函数名 | **px\_void MP\_ErrorCatch(px\_memorypool \*Pool,PX\_MP\_ErrorCall ErrorCall);** |
| 说明 | **设置内存池的错误回调函数** |
| 参数 | Pool：指向MemoryPool的结构体，这个结构体使用MP\_Create进行创建  ErrorCall：异常回调函数 |
| 返回值 | - |

**typedef px\_void (\*PX\_MP\_ErrorCall)(PX\_MEMORYPOOL\_ERROR);**

**当内存池触发异常时将会执行该异常回调函数,如果没有设置异常回调将会以PX\_ERROR进行处理**

**异常回调消息包括**

**PX\_MEMORYPOOL\_ERROR\_OUTOFMEMORY 内存池内存不足**

**PX\_MEMORYPOOL\_ERROR\_INVALID\_ACCESS 内存池指向内存无法读/写访问**

**PX\_MEMORYPOOL\_ERROR\_INVALID\_ADDRESS 释放无效的内存地址**

### 字符串操作

##### 字符串拷贝函数

dst:指向目标内存

src:指向源字符串

size:目标内存的大小

void px\_strcpy(px\_char \*dst,px\_char \*src,px\_uint size);

##### 宽字符串拷贝函数

dst:指向目标内存

src:指向源字符串

size:目标内存的大小

void px\_wstrcpy(px\_word \*dst,px\_ word \*src,px\_uint size);

##### 字符串长度

|  |  |
| --- | --- |
| 函数名 | px\_int px\_strlen(px\_char \*dst) |
| 说明 | **字符串长度** |
| 参数 | 字符串指针 |
| 返回值 | 返回长度 |

##### 宽字符字符串长度

|  |  |
| --- | --- |
| 函数名 | px\_int px\_wstrlen(px\_word \*dst) |
| 说明 | **宽字符字符串长度** |
| 参数 | 字符串指针 |
| 返回值 | 返回长度 |

##### 字符串设置

|  |  |
| --- | --- |
| 函数名 | px\_void px\_strset(px\_char \*dst, const px\_char \*src); |
| 说明 | **将目标字符串拷贝为源字符串** |
| 参数 | Dst 目标字符串原指针  src 源字符串 |
| 返回值 | - |

##### 字符串拼接

|  |  |
| --- | --- |
| 函数名 | px\_void px\_strcat(px\_char \*src, const px\_char \*cat); |
| 说明 | **字符串拼接** |
| 参数 | src字符串原指针  cat 需要拼接到src的字符串 |
| 返回值 | - |

##### 宽字符串拼接

|  |  |
| --- | --- |
| 函数名 | px\_void px\_wstrcat(px\_word \*src,const px\_ word \*cat); |
| 说明 | **宽字符串拼接** |
| 参数 | src字符串原指针  cat 需要拼接到src的字符串 |
| 返回值 | - |

##### 字符串转大写

|  |  |
| --- | --- |
| 函数名 | px\_void px\_strupr(px\_char \*src); |
| 说明 | **将src指向字符串转换为大写** |
| 参数 | src字符串原指针 |
| 返回值 | - |

##### 字符串转小写

|  |  |
| --- | --- |
| 函数名 | px\_void px\_strlwr(px\_char \*src); |
| 说明 | **将src指向字符串转换为小写** |
| 参数 | src字符串原指针 |
| 返回值 | - |

##### 字符串全等

|  |  |
| --- | --- |
| 函数名 | px\_bool px\_strequ(px\_char \*src,px\_char \*dst); |
| 说明 | **字符串全等** |
| 参数 | src字符串指针  dst 字符串指针 |
| 返回值 | 若两个字符串全等,返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool px\_strequ2(px\_char \*src,px\_char \*dst); |
| 说明 | **字符串全等(忽略大小写)** |
| 参数 | src字符串指针  dst 字符串指针 |
| 返回值 | 若两个字符串全等(忽略大小写),返回PX\_TRUE,否者返回PX\_FALSE |

##### 字符串是否是数字

|  |  |
| --- | --- |
| 函数名 | px\_bool px\_strIsNumeric(px\_char \*src); |
| 说明 | **字符串是否是数字** |
| 参数 | src字符串指针 |
| 返回值 | 若字符串是数字,返回PX\_TRUE,否者返回PX\_FALSE |

##### 字符串是否是小数

|  |  |
| --- | --- |
| 函数名 | px\_bool px\_strIsFloat(px\_char \*src); |
| 说明 | **字符串是否是小数** |
| 参数 | src字符串指针 |
| 返回值 | 若字符串是小数,返回PX\_TRUE,否者返回PX\_FALSE |

##### 字符串是否是整数

|  |  |
| --- | --- |
| 函数名 | px\_bool px\_strIsInt(px\_char \*src); |
| 说明 | **字符串是否是整数** |
| 参数 | src字符串指针 |
| 返回值 | 若字符串是整数,返回PX\_TRUE,否者返回PX\_FALSE |

##### 字符串查找字符

|  |  |
| --- | --- |
| 函数名 | px\_char \*PX\_strchr(const char \*s,int ch); |
| 说明 | **字符串查找字符** |
| 参数 | s查找字符串,ch 字符 |
| 返回值 | 如果找到返回首指针,否者返回PX\_NULL |

##### 字符串查找字符串

|  |  |
| --- | --- |
| 函数名 | char\* PX\_strstr(const char\* dest, const char\* src); |
| 说明 | **字符串查找字符串** |
| 参数 | dest 待比较字符串  src 比对字符串 |
| 返回值 | 如果找到返回首指针,否者返回PX\_NULL |

##### 字符串格式化

|  |  |
| --- | --- |
| 函数名 | px\_int px\_sprintf1(px\_char \*\_out\_str,px\_int str\_size,px\_char fmt[], px\_stringformat \_1)  px\_int px\_sprintf2(px\_char \*\_out\_str,px\_int str\_size,px\_char fmt[], px\_stringformat \_1, px\_stringformat \_2)  px\_int px\_sprintf2(px\_char \*\_out\_str,px\_int str\_size,px\_char fmt[], px\_stringformat \_1, px\_stringformat \_2, px\_stringformat \_3)  ….. |
| 说明 | **字符串格式化函数** |
| 参数 | \_out\_str输出字符串指针  Fmt 格式化输出字符串  使用%1 %2 %3….来表示需要格式化的字符串  当需要指定浮点精度时,可以使用%1.x的格式来表述,其中x为拓展精度  \_1,\_2,\_3…..参数,数量和px\_sprintf后的数字对应,用于对应%1 %2…对应格式化字符串  px\_stringformat可以用  PX\_STRING\_FORMAT\_INT  PX\_STRING\_FORMAT\_FLOAT  PX\_STRING\_FORMAT\_STRING  函数,进行构造分别用于表述一个整数,浮点数,字符串 |
| 返回值 | - |

##### \*取得文件路径的文件名

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_FileGetName(const px\_char filefullName[],px\_char \_out[],px\_int outSize); |
| 说明 | 取得文件路径的文件名 |
| 参数 | filefullName路径  out 输出文件名  outsize 输出数组长度 |
| 返回值 |  |

##### \*取得文件路径的路径

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_FileGetPath(const px\_char filefullName[],px\_char \_out[],px\_int outSize); |
| 说明 | 取得文件路径的路径 |
| 参数 | filefullName路径  out 输出路径  outsize 输出数组长度 |
| 返回值 |  |

##### \*取得文件路径的后缀名

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_FileGetExt(const px\_char filefullName[],px\_char \_out[],px\_int outSize); |
| 说明 | 取得文件路径的后缀名 |
| 参数 | filefullName路径  out 输出路径  outsize 输出数组长度 |
| 返回值 |  |

### 转换

#### 字节类型转十六进制字符串

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_BufferToHexString(px\_byte data[],px\_int size,px\_char hex\_str[]); |
| 说明 | **数据类型转十六进制字符串类型** |
| 参数 | data数据指针,size为数据长度,hex\_str字符串指针长度为size\*2+1, |
| 返回值 |  |

#### 十六进制字符串转字节类型

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_HexStringToBuffer(const px\_char hex\_str[],px\_byte data[]) |
| 说明 | **十六进制字符串转数据类型** |
| 参数 | hex\_str字符串指针,data数据指针,长度是hex\_str字符串长度的一半 |
| 返回值 | 成功返回PX\_TRUE,失败返回PX\_FALSE; |

#### 十六进制字符转数字

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_htoi(char\* ) |
| 说明 | **十六进制字符串转px\_uint** |
| 参数 | 字符串指针 |
| 返回值 | 返回转换值 |

#### 字符转数字

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_atoi(char\* ) |
| 说明 | **字符串转px\_int** |
| 参数 | 字符串指针 |
| 返回值 | 返回转换值 |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_atof(char\* ) |
| 说明 | **字符串转px\_float** |
| 参数 | 字符串指针 |
| 返回值 | 返回转换值 |

#### 字符串转IPv4地址

|  |  |
| --- | --- |
| 函数名 | px\_dword PX\_inet\_addr(px\_char i[]); |
| 说明 | **字符串转IPv4地址** |
| 参数 | I字符串ip地址 |
| 返回值 | 转换后的双字 |

#### IPv4地址转字符串

|  |  |
| --- | --- |
| 函数名 | px\_char\* PX\_inet\_ntoa(px\_dword ipv4); |
| 说明 | **IPv4转字符串地址** |
| 参数 | Ipv4双字ip地址 |
| 返回值 | 转换后的字符串 |

#### 数字转字符串

整数

|  |  |
| --- | --- |
| 函数名 | px\_int \*PX\_itoa(int num,char \*str,px\_int size,int radix) |
| 说明 | **数字转字符串** |
| 参数 | num 数字  str 结果字符串指针  size 缓存大小  radix 进制 |
| 返回值 | 返回转换长度 |

|  |  |
| --- | --- |
| 函数名 | PX\_RETRUN\_STRING PX\_itos(px\_int num,px\_int radix); |
| 说明 | **数字转字符串** |
| 参数 | num 数字  radix 进制 |
| 返回值 | 返回一个字符串结构体,访问其data成员可直接取得转换字符串 |

小数

|  |  |
| --- | --- |
| 函数名 | px\_int \*PX\_ftoa(float num,char \*str,px\_int size,px\_int precision) |
| 说明 | **数字转字符串** |
| 参数 | num 数字  str 结果字符串指针  size 缓存大小  precision精度长度 |
| 返回值 | 返回转换长度 |

|  |  |
| --- | --- |
| 函数名 | PX\_RETRUN\_STRING PX\_ftos(float f, int precision); |
| 说明 | **数字转字符串** |
| 参数 | num 数字  precision精度长度 |
| 返回值 | 返回一个字符串结构体,访问其data成员可直接取得转换字符串 |

#### 弧度角度转换

|  |  |
| --- | --- |
| 函数名 | PX\_RadianToAngle(angle) |
| 说明 | **弧度转换为角度** |
| 参数 | angle 角度 |
| 返回值 | 角度 |

|  |  |
| --- | --- |
| 函数名 | PX\_AngleToRadian(radian) |
| 说明 | **角度转弧度** |
| 参数 | radian 弧度 |
| 返回值 | 弧度 |

### 字节序

#### 主机字节序转网络字节序

|  |  |
| --- | --- |
| 函数名 | px\_dword PX\_htonl(px\_dword h);  px\_word PX\_htons(px\_word h); |
| 说明 | **主机字节序转网络字节序** |
| 参数 | h:dword或word类型值 |
| 返回值 | 转换后的字节序 |

#### 网络字节序转主机字节序

|  |  |
| --- | --- |
| 函数名 | px\_dword PX\_ntohl(px\_dword n);  px\_word PX\_ntohs(px\_word n); |
| 说明 | **网络字节序转主机字节序** |
| 参数 | h:dword或word类型值 |
| 返回值 | 转换后的字节序 |

### 数学

#### 素数模乘同余发生器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_srand(px\_uint64 seed); |
| 说明 | **初始化一个素数模乘同余发生器的随机数种子** |
| 参数 | seed 种子,如果未调用该函数,将会以默认值对其初始化 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_rand(); |
| 说明 | **返回一个0~PX\_RAND\_MAX-1之间的随机数** |
| 参数 |  |
| 返回值 | 返回随机数,范围为0~PX\_RAND\_MAX-1 |

|  |  |
| --- | --- |
| 函数名 | px\_uint32 PX\_randEx(px\_uint64 seed) |
| 说明 | **返回一个0~PX\_RAND\_MAX-1之间的随机数以seed作为运算种子** |
| 参数 |  |
| 返回值 | 返回随机数,范围为0~PX\_RAND\_MAX-1 |

#### 高斯随机数发生器



|  |  |
| --- | --- |
| 函数名 | px\_int PX\_GaussRand(); |
| 说明 | **返回一个符合高斯分布的随机数** |
| 参数 |  |
| 返回值 | 返回高斯随机数,见上图分布 |

#### 对数函数

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_ln(px\_double \_\_x)  px\_double PX\_log(px\_double \_\_x) |
| 说明 | **求一个数e为底对数函数** |
| 参数 | \_\_x这个数的值 |
| 返回值 |  |

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_lg(px\_double \_\_x)  px\_double PX\_log10(px\_double \_\_x) |
| 说明 | **求一个数10为底对数函数** |
| 参数 | \_\_x这个数的值 |
| 返回值 |  |

#### 三角函数

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_sind(px\_ double radian);  px\_double PX\_cosd(px\_ double radian); |
| 说明 | **px\_double PX\_sind(px\_ double** radian**); 弧度制求正弦**  **px\_double PX\_cosd(px\_ double** radian**); 弧度制求余弦**  **该函数使用泰勒展开** |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_sin\_radian(px\_float radian);  px\_float PX\_cos\_radian(px\_float radian);  px\_float PX\_tan\_radian(px\_float radian);  px\_float PX\_sin\_angle(px\_float angle);  px\_float PX\_cos\_angle(px\_float angle);  px\_float PX\_tan\_angle(px\_float angle);  px\_double PX\_atan(px\_double x);  px\_double PX\_atan2(px\_double y, px\_double x);  px\_double PX\_asin(px\_double x);  px\_double PX\_acos(px\_double x); |
| 说明 | **px\_float PX\_sin\_radian(px\_float** radian**); 弧度制求正弦**  **px\_float PX\_cos\_radian(px\_float** radian**); 弧度制求余弦**  **px\_float PX\_tan\_radian(px\_float** radian**); 弧度制求正切**  **px\_float PX\_sin\_angle(px\_float angle);角度制求正弦**  **px\_float PX\_cos\_angle(px\_float angle); 角度制求余弦**  **px\_float PX\_tan\_angle(px\_float angle); 角度制求正切**  **px\_double PX\_atan(px\_double x); 弧度制求反正切**  **px\_double PX\_atan2(px\_double y, px\_double x); 弧度制atan2**  **px\_double PX\_asin(px\_double x); 弧度制arcsin**  **px\_double PX\_acos(px\_double x); 弧度制arccos** |

#### 平方根

|  |  |
| --- | --- |
| 函数名 | float PX\_Sqrt( float number ) |
| 说明 | **求一个数的平方根** |
| 参数 | Number 需要求平方根的数 |
| 返回值 | 返回这个数的平方根（这个数必须是正数） |

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_Sqrtd(px\_double number ) |
| 说明 | **求一个数更高精度的平方根** |
| 参数 | Number 需要求平方根的数 |
| 返回值 | 返回这个数的平方根（这个数必须是正数） |

|  |  |
| --- | --- |
| 函数名 | float PX\_SqrtRec( float number ) |
| 说明 | **求一个数平方根的倒数** |
| 参数 | Number 需要求平方根的倒数 |
| 返回值 | 返回这个数的平方根的倒数（这个数必须是正数） |

#### 方差

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_Variance(px\_double x[],px\_int n) |
| 说明 | **计算一个序列的方差** |
| 参数 | x 序列数据  n 元素个数 |
| 返回值 | 返回这个矩阵计算方差 |

#### 协方差

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_Covariance(px\_double x[],px\_double y[],px\_int n); |
| 说明 | **计算一个矩阵的协方差** |
| 参数 | x, y两数据  n 数据长度 |
| 返回值 | 返回这两个数据协计算方差 |

#### 求绝对值

|  |  |
| --- | --- |
| 函数名 | PX\_ABS(x) |
| 说明 | **宏函数求绝对值** |
| 参数 | 值 |
| 返回值 | 返回绝对值 |

#### tanh

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_tanh(px\_double x) |
| 说明 | **tanh函数** |
| 参数 | x |
| 返回值 | tanh计算结果 |

#### sigmoid

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_ sigmoid(px\_double x) |
| 说明 | **sigmoid函数** |
| 参数 | x |
| 返回值 | **sigmoid**计算结果 |

#### ReLU

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_ sigmoid(px\_double x) |
| 说明 | **ReLU函数** |
| 参数 | x |
| 返回值 | ReLU计算结果 |

#### 四舍五入

|  |  |
| --- | --- |
| 函数名 | PX\_APO(x) |
| 说明 | **宏函数小数点后四舍五入求整数** |
| 参数 | 小数 |
| 返回值 | 返回四舍五入的整数 |

#### 幂

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_pow\_ii(px\_int i,px\_int n); |
| 说明 | **整数整幂** |
| 参数 | i底数 n次幂 |
| 返回值 | i的n幂 |

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_pow\_ff(double num,double m); |
| 说明 | **幂** |
| 参数 | num底数 m次幂 |
| 返回值 | num的m幂 |

#### 复数

|  |  |
| --- | --- |
| 函数名 | px\_complex PX\_complexBuild(px\_float re,px\_float im); |
| 说明 | **构造一个复数** |
| 参数 | re为实部,im为虚部 |
| 返回值 | 构造的复数结构 |

|  |  |
| --- | --- |
| 函数名 | px\_complex PX\_complexAdd(px\_complex a,px\_complex b); |
| 说明 | **复数相加** |
| 参数 | a,b 需要相加的两复数 |
| 返回值 | 结果 |

|  |  |
| --- | --- |
| 函数名 | px\_complex PX\_complexMult(px\_complex a,px\_complex b); |
| 说明 | **复数相乘** |
| 参数 | a,b 需要相乘的两复数 |
| 返回值 | 结果 |

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_complexMod(px\_complex a) |
| 说明 | **复数的模** |
| 参数 | 需要求解的复数 |
| 返回值 | 结果 |

|  |  |
| --- | --- |
| 函数名 | px\_complex PX\_complexLog(px\_complex a); |
| 说明 | **复数的e的对数** |
| 参数 | 需要求解的复数 |
| 返回值 | 结果 |

|  |  |
| --- | --- |
| 函数名 | px\_complex PX\_complexExp(px\_complex a); |
| 说明 | **复数的e的次数** |
| 参数 | 需要求解的复数 |
| 返回值 | 结果 |
| 函数名 | px\_complex PX\_complexSin(px\_complex a); |
| 说明 | 复数的sin值 |
| 参数 | 需要求解的复数 |
| 返回值 | 结果 |

#### 矩阵

Px\_matrix 结构体，声明为

union {

struct {

px\_float \_11, \_12, \_13, \_14;

px\_float \_21, \_22, \_23, \_24;

px\_float \_31, \_32, \_33, \_34;

px\_float \_41, \_42, \_43, \_44;

};

px\_float m[4][4];

}

是一个4x4的矩阵

|  |  |
| --- | --- |
| 函数名 | void PX\_MatrixZero(px\_matrix \*Mat); |
| 说明 | **构造一个0矩阵** |
| 参数 | Mat指向该矩阵的指针 |

|  |  |
| --- | --- |
| 函数名 | void PX\_MatrixIdentity(px\_matrix \*Mat); |
| 说明 | **构造一个单位矩阵** |
| 参数 | Mat指向该矩阵的指针 |

|  |  |
| --- | --- |
| 函数名 | px\_matrix PX\_MatrixMultiply(px\_matrix Mat1,px\_matrix Mat2); |
| 说明 | **矩阵相乘** |
| 参数 | Mat1，Mat2 需要相乘的两个矩阵 |
| 返回值 | 返回矩阵相乘结果矩阵 |

|  |  |
| --- | --- |
| 函数名 | px\_matrix PX\_MatrixAdd(px\_matrix Mat1,px\_matrix Mat2); |
| 说明 | **矩阵相加** |
| 参数 | Mat1，Mat2 需要相加的两个矩阵 |
| 返回值 | 返回矩阵相加结果矩阵 |

|  |  |
| --- | --- |
| 函数名 | px\_matrix PX\_MatrixSub(px\_matrix Mat1,px\_matrix Mat2); |
| 说明 | **矩阵相减** |
| 参数 | Mat1，Mat2 需要相减的两个矩阵 |
| 返回值 | 返回矩阵相减结果矩阵 |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MatrixEqual(px\_matrix Mat1,px\_matrix Mat2); |
| 说明 | **判断两个矩阵是否相等** |
| 参数 | Mat1，Mat2 需要判断的两个矩阵 |
| 返回值 | 如果相等返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixTranslation(px\_matrix \*mat,float x,float y,float z); |
| 说明 | **构造平移矩阵** |
| 参数 | Mat指向该矩阵的指针  X,Y,Z平移向量 |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixRotateX(px\_matrix \*mat,float Angle); |
| 说明 | **构造X轴旋转矩阵** |
| 参数 | Mat指向该矩阵的指针  Angle 旋转角度（角度制） |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixRotateY(px\_matrix \*mat,float Angle); |
| 说明 | **构造Y轴旋转矩阵** |
| 参数 | Mat指向该矩阵的指针  Angle 旋转角度（角度制） |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixRotateZ(px\_matrix \*mat,float Angle); |
| 说明 | **构造Z轴旋转矩阵** |
| 参数 | Mat指向该矩阵的指针  Angle 旋转角度（角度制） |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixRotateXRadian (px\_matrix \*mat,float rad); |
| 说明 | **构造X轴旋转矩阵** |
| 参数 | Mat指向该矩阵的指针  rad 旋转角度（弧度制） |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixRotateYRadian (px\_matrix \*mat,float rad); |
| 说明 | **构造Y轴旋转矩阵** |
| 参数 | Mat指向该矩阵的指针  rad 旋转角度（弧度制） |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixRotateZRadian(px\_matrix \*mat,float rad); |
| 说明 | **构造Z轴旋转矩阵** |
| 参数 | Mat指向该矩阵的指针  rad 旋转角度（弧度制） |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixScale(px\_matrix \*mat,float x,float y,float z); |
| 说明 | **构造缩放矩阵** |
| 参数 | Mat指向该矩阵的指针  X,Y,Z缩放方向 |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MatrixInverse(px\_matrix \*mat); |
| 说明 | **矩阵求逆** |
| 参数 | Mat指向该矩阵的指针  如果求逆成功，则mat为逆矩阵 |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MatrixTranspose(px\_matrix \*matrix); |
| 说明 | **矩阵转置** |
| 参数 | Marixt为需要转置的矩阵指针 |

|  |  |
| --- | --- |
| 函数名 |  |
| 说明 |  |
| 参数 |  |

#### 点/向量

|  |  |
| --- | --- |
| 函数名 | px\_point2D PX\_Point2DRotate(px\_point2D p,px\_float angle);  px\_point PX\_PointRotate(px\_point p,px\_float angle); |
| 说明 | **返回一个点绕原点顺时针旋转角度后的点** |
| 参数 | P绕原点旋转的参考点  angle 旋转角度 |
| 返回值 | 返回结果点 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointAdd(px\_point p1,px\_point p2);  px\_point PX\_Point2DAdd(px\_point2D p1,px\_point2D p2);  px\_point PX\_Point4DAdd(px\_point4D p1,px\_point4D p2); |
| 说明 | **向量相加** |
| 参数 | P1,p2需要相加的两个向量 |
| 返回值 | 返回结果向量 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointSub(px\_point p1,px\_point p2);  px\_point2D PX\_Point2DSub(px\_point2D p1,px\_point2D p2);  px\_point4D PX\_Point4DSub(px\_point4D p1,px\_point4D p2); |
| 说明 | **向量相减** |
| 参数 | P1,p2需要相减的两个向量 |
| 返回值 | 返回结果向量 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointMul(px\_point p1,px\_float m);  px\_point2D PX\_Point2DMul(px\_point2D p1,px\_float m);  px\_point4D PX\_Point4DMul(px\_point4D p1,px\_float m); |
| 说明 | **向量缩放** |
| 参数 | P1 缩放向量  M 倍数 |
| 返回值 | 返回结果向量 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointDiv(px\_point p1,px\_float m);  px\_point2D PX\_Point2DDiv(px\_point2D p1,px\_float m);  px\_point4D PX\_PointDiv(px\_point4D p1,px\_float m); |
| 说明 | **向量缩放** |
| 参数 | P1 缩放向量  M 除以倍数 |
| 返回值 | 返回结果向量 |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_PointDot(px\_point p1,px\_point p2);  px\_float PX\_Point2DDot(px\_point2D p1,px\_point2D p2);  px\_float PX\_Point4DDot(px\_point4D p1,px\_point4D p2); |
| 说明 | **向量点乘** |
| 参数 | P1,p2两点乘向量 |
| 返回值 | 返回结果 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointCross(px\_point p1,px\_point p2);  px\_point2D PX\_Point2DCross(px\_point2D p1,px\_point2D p2);  px\_point4D PX\_Point4DCross(px\_point4D p1,px\_point4D p2); |
| 说明 | **向量叉乘** |
| 参数 | P1,P2 两叉乘向量 |
| 返回值 | 返回叉乘结果 |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_PointMod(px\_point p);  px\_float PX\_Point2DMod(px\_point2D p);  px\_float PX\_Point4DMod(px\_point4D p); |
| 说明 | **向量的模** |
| 参数 | P 该向量 |
| 返回值 | 返回该向量的模 |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_PointSquare(px\_point p); |
| 说明 | **向量模的平方** |
| 参数 | P 该向量 |
| 返回值 | 返回该向量模的平方 |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_PointDistance(px\_point p1,px\_point p2); |
| 说明 | **求两点间距离** |
| 参数 | P1 p2 两点 |
| 返回值 | 返回两点间距离 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointNormalization(px\_point p);  px\_point2D PX\_Point2DNormalization(px\_point p);  px\_point4D PX\_Point4DNormalization(px\_point p); |
| 说明 | **求单位向量(归一化向量)** |
| 参数 | P 该向量 |
| 返回值 | 返回该向量模的单位向量 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointMulMatrix(px\_point p,px\_matrix mat)  px\_point2D PX\_Point2DMulMatrix(px\_point p,px\_matrix mat) |
| 说明 | **点乘矩阵** |
| 参数 | P 该向量，mat变换矩阵 |
| 返回值 | 返回变换后的点 |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_Point\_sin (px\_point p) |
| 说明 | **求该向量与x轴夹角sin值** |
| 参数 | P 该向量 |
| 返回值 | Sin值 |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_Point\_cos (px\_point p) |
| 说明 | **求该向量与x轴夹角cos值** |
| 参数 | P 该向量 |
| 返回值 | cos值 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointReflectX(px\_point vector\_refer,px\_point respoint) |
| 说明 | **参照向量关于x轴夹角对一个点做相应旋转** |
| 参数 | vector\_refer 向量  respoint 原点 |
| 返回值 | 旋转后的点 |

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_PointInverse(px\_point p1) |
| 说明 | **向量方向取反** |
| 参数 | P1 需要取反的向量 |
| 返回值 | 取反的向量 |

#### 交叉碰撞

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_isLineCrossRect(px\_point p1,px\_point p2,px\_rect rect,px\_point \*cp1,px\_point \*cp2); |
| 说明 | **检测一条线与矩形是否相交** |
| 参数 | P1,p2 这条线的两个点  Rect 矩形范围  Cp1 如果有相交,输出交点1  Cp2 如果有相交,输出交点2 |
| 返回值 | 如果有交叉返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_isRectCrossRect(px\_rect rect1,px\_rect rect2); |
| 说明 | **检测两个矩形是否有交叉碰撞** |
| 参数 | Rect1  Rect2  两矩形范围 |
| 返回值 | 如果有交叉返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_isRectCrossCircle(px\_rect rect1,px\_point center,px\_float radius); |
| 说明 | **检测一个矩形和一个圆是否有交叉碰撞** |
| 参数 | Rect1 矩形  Center 圆心  Radius 圆半径 |
| 返回值 | 如果有交叉返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_isCircleCrossCircle(px\_point center1,px\_float radius1,px\_point center2,px\_float radius2); |
| 说明 | **检测两个圆是否有交叉碰撞** |
| 参数 | Center1 Center2圆心1,2  Radius1 Radius2圆1,2半径 |
| 返回值 | 如果有交叉返回PX\_TRUE,否者返回PX\_FALSE |

#### 快速排序(由小到大)

|  |  |
| --- | --- |
| 函数名 | void PX\_Quicksort\_MinToMax(PX\_QuickSortAtom array[], px\_int left, px\_int right); |
| 说明 | **对特定结构体进行由小到大排序** |
| 参数 | typedef struct  {  px\_int power;//决定排序的权值  px\_void \*pData;  }PX\_QuickSortAtom;  array 排序序列  left 开始索引  right 结束索引 |

#### 快速排序(由大到小)

|  |  |
| --- | --- |
| 函数名 | void PX\_Quicksort\_MaxToMin(PX\_QuickSortAtom array[], px\_int left, px\_int right); |
| 说明 | **对特定结构体进行由大到小排序** |
| 参数 | typedef struct  {  px\_int power;//决定排序的权值  px\_void \*pData;  }PX\_QuickSortAtom;  array 排序序列  left 开始索引  right 结束索引 |

#### 矩形

px\_rect

typedef struct \_px\_rect

{

px\_float x,y,width,height;

}px\_rect;

x,y为矩形左上角坐标，width与height为宽高

|  |  |
| --- | --- |
| 函数名 | px\_rect PX\_RECT(px\_float x,px\_float y,px\_float width,px\_float height); |
| 说明 | **构造一个矩形** |
| 参数 | x,y 左上角坐标  width,height 矩形的宽度和高度 |
| 返回值 | 返回构造的矩形结构体 |

|  |  |
| --- | --- |
| 函数名 | px\_rect PX\_RECTPOINT2(px\_point p1,px\_point p2); |
| 说明 | **用两个点构造一个矩形** |
| 参数 | p1,p2 构造的两点 |
| 返回值 | 返回构造的矩形结构体 |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_isPointInRect(px\_point p,px\_rect rect) |
| 说明 | **判断点是否在矩形内** |
| 参数 | p该点，rect为该矩形 |
| 返回值 | 若在矩形内返回PX\_TRUE 否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_isRectInRect(px\_rect rect1,px\_rect rect2) |
| 说明 | **判断是两个矩形是否相交** |
| 参数 | Rect1 矩形1,rect2 矩形2 |
| 返回值 | 若在矩形内返回PX\_TRUE 否者返回PX\_FALSE |

#### CRC校验（CRC32）

|  |  |
| --- | --- |
| 函数名 | px\_uint32 PX\_crc32( px\_byte \*buffer, px\_uint size); |
| 说明 | **对一段缓存区进行CRC32校验** |
| 参数 | Buffer 缓存区  Size 缓存区大小 |
| 返回值 | CRC32校验值 |

#### CRC校验（CRC16）

|  |  |
| --- | --- |
| 函数名 | px\_uint16 PX\_crc16( px\_byte \*buffer, px\_uint size); |
| 说明 | **对一段缓存区进行CRC16校验** |
| 参数 | Buffer 缓存区  Size 缓存区大小 |
| 返回值 | CRC32校验值 |

#### 累加和校验（Sum32）

|  |  |
| --- | --- |
| 函数名 | px\_uint32 PX\_sum32( px\_byte \*buffer, px\_uint size); |
| 说明 | **对一段缓存区进行累加和校验** |
| 参数 | Buffer 缓存区  Size 缓存区大小 |
| 返回值 | 累加和校验值 |

#### 第一类修正贝塞尔函数

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_Bessel(int n,double x); |
| 说明 | **第一类修正贝塞尔函数** |
| 参数 | n,x |
| 返回值 | 对应PX\_Bessel(0,x), PX\_Bessel(1,x), PX\_Bessel(2,x)….. |

### 通讯链接器(IO虚拟层)

通讯链接器(PX\_Linker)是PainterEngine默认的对通讯协议/IO的虚拟层,它可以被解释为TCP,UDP,串口,并口,文件读写…等通讯协议的IO统一及虚拟

#### 初始化链接器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LinkerInitialize(PX\_Linker \*linker,PX\_LinkerRead read,px\_void \*readUserPtr,PX\_LinkerWrite write,px\_void \*writeUserPtr); |
| 说明 | **初始化链接器** |
| 参数 | linker 链接器结构体  read 读函数(需要自行实现)  readuserptr 读函数传递指针  write 写函数(需要自行实现)  writeuserptr 写函数传递指针  \*注意,对于有连接协议的抽象,在读写函数中必须自行维护连接的可用性 |
| 返回值 | - |

#### 链接器读

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_LinkerRead(PX\_Linker \*linker,px\_void \*data,px\_int datasize); |
| 说明 | **链接器读** |
| 参数 | linker 链接器结构体  data 读缓存  datasize读缓存大小 |
| 返回值 | 成功读字节数 |

#### 链接器写

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_LinkerWrite(PX\_Linker \*linker,px\_void \*data,px\_int datasize); |
| 说明 | **链接器写** |
| 参数 | linker 链接器结构体  data 写缓存  datasize写缓存大小 |
| 返回值 | 成功写字节数 |

### 数据结构

#### 取得结构体偏移量

|  |  |
| --- | --- |
| 函数名 | PX\_STRUCT\_OFFSET(t,m) |
| 说明 | **取得结构体成员m对于结构体的偏移量** |
| 参数 | t 结构体类型  m 成员名 |
| 返回值 | 一个px\_uint类型,为m在该结构体的偏移量 |

#### 字母

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_CharIsNumeric(px\_char ch) |
| 说明 | **判断字母是否是数字** |
| 参数 | ch字母 |
| 返回值 | 若是返回PX\_TRUE,若不是返回PX\_FALSE |

#### 链表

PainterEngine提供链表模板函数px\_list，对该链表使用前都要改由PX\_ListInitialize进行初始化并在使用完毕后使用PX\_ListFree释放内存

##### 初始化

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ListInitialize (px\_list \*list,px\_memorypool \*mp); |
| 说明 | **对链表结构进行初始化** |
| 参数 | list 需要初始化的结构指针  mp 链表部署内存池指针 |

##### 插入数据

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ListPush(px\_list \*list,px\_void \*data,px\_int size); |
| 说明 | **插入一个数据到链表** |
| 参数 | list 结构指针  data 数据指针  size 数据大小 |

##### 删除数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ListPop(px\_list \*list,px\_list\_node \*node); |
| 说明 | **将一个节点为node的数据从链表中删除** |
| 参数 | list 结构指针  node 链表节点 |
| 返回值 | 若成功删除返回PX\_TRUE，否者为PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ListErase(px\_list \*list,px\_int i); |
| 说明 | **将一个节点为node的数据从链表中删除** |
| 参数 | list 结构指针  i 节点索引 |
| 返回值 | 若成功删除返回PX\_TRUE，否者为PX\_FALSE |

##### 取得链表节点

|  |  |
| --- | --- |
| 函数名 | px\_list\_node\* PX\_ListNodeAt(px\_list \*list,px\_int index); |
| 说明 | **取得链表当前节点指针** |
| 参数 | list结构指针  index 索引 |
| 返回值 | 索引节点,如果索引不合法返回PX\_NULL |

##### 取得下一个链表节点

|  |  |
| --- | --- |
| 函数名 | px\_list\_node\* PX\_ListNodeNext(px\_list\_node\* node); |
| 说明 | **取得节点的下一个节点指针** |
| 参数 | node当前链表节点 |
| 返回值 | 下一个链表节点指针 |

##### 取得链表当前节点数

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_ListSize(px\_list \*list); |
| 说明 | **取得链表当前节点数** |
| 参数 | list结构指针 |
| 返回值 | 当前节点数 |

##### 移动节点

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ListMove(px\_list \*list,px\_int index,px\_int moveto); |
| 说明 | **移动一个索引节点到目标索引** |
| 参数 | index要移动的节点索引  moveto 移动到的索引 |
| 返回值 |  |

##### 取得链表节点数

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_ListSize(px\_list \*list); |
| 说明 | **取得链表节点数** |
| 参数 | list结构指针 |
| 返回值 | 链表节点数 |

##### 清空链表（释放占用内存）

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ListClear(px\_list \*list); |
| 说明 | **清空链表** |
| 参数 | list结构指针 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ListFree(px\_list \*list); |
| 说明 | **清空链表** |
| 参数 | list结构指针 |
| 返回值 | - |

##### 取节点数据

|  |  |
| --- | --- |
| 函数名 | PX\_LIST\_NODEDATA(x)  PX\_LIST\_NODETDATA (t,x) ((t \*)(x->pdata))  #define PX\_LISTAT(t,x,i) ((t \*)(PX\_ListNodeAt(x,i)->pdata)) |
| 说明 | **取节点数据** |
| 参数 | x结构指针  \*t 类型 |
| 返回值 | - |

##### DEMO

px\_list list;

PX\_ListInitialize(&list,&mempool2);

PX\_ListPush(&list,"Hello",6);

PX\_ListPush(&list,"World",6);

PX\_ListPush(&list,"binar",6);

PX\_ListPop(&list,PX\_ListAt(&list,1));

PX\_ListPop(&list,PX\_ListAt(&list,1));

PX\_ListPop(&list,PX\_ListAt(&list,1));

for (int i=0;i<PX\_ListSize(&list);i++)

{

printf((px\_char \*)PX\_LIST\_NODEDATA(PX\_ListAt(&list,i)));

}

#### 容器

PainterEngine提供容器模板函数px\_vector，对该链表使用前都要改由PX\_VectorInitialize进行初始化并在使用完毕后使用PX\_VectorFree释放内存

##### 初始化

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_VectorInitialize  (px\_vector \*vec,px\_memorypool \*mp,px\_int nodeSize,px\_int init\_size); |
| 说明 | **对容器结构进行初始化** |
| 参数 | vec 容器指针  mp 容器部署内存池指针  nodesize 每个元素的大小  init\_size 初始容器大小(初始化分配的元素个数) |

##### 添加元素

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_VectorPushback(px\_vector \*vec,px\_void \*data); |
| 说明 | **在容器中添加一个元素** |
| 参数 | vec容器指针  data 压入数据指针 |
| 返回值 | 成功返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_VectorPushTo(px\_vector \*vec,px\_void \*data,px\_int index); |
| 说明 | **在容器中指定位置添加一个元素** |
| 参数 | vec容器指针  data 压入数据指针  index 插入的位置索引 |
| 返回值 | 成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 设定元素

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_VectorSet(px\_vector \*vec,px\_uint index,px\_void \*data); |
| 说明 | **在容器中设定对应索引的元素,注意,如果索引小于分配容量,改容器将会重新分配内存空间以设定元素,同时索引小于使用大小,容器也会设定对应使用大小以适应设定索引** |
| 参数 | vec容器指针  index 索引  data 压入数据指针 |
| 返回值 | 成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 删除元素

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_VectorErase(px\_vector \*vec,px\_int index); |
| 说明 | **在容器中删除一个元素** |
| 参数 | vec容器指针  index 需要删除的索引 |
| 返回值 | 若成功返回PX\_TRUE，否者PX\_FALSE |

##### 清空容器(不释放预留内存)

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_VectorClear(px\_vector \*vec); |
| 说明 | **清空容器** |
| 参数 | vec容器指针 |
| 返回值 | - |

##### 取容器数据

|  |  |
| --- | --- |
| 函数名 | PX\_VectorAt(vec,i) |
| 说明 | **取容器数据** |
| 参数 | vec容器指针  I 索引 |
| 返回值 | - |

##### 释放容器(释放预留内存)

|  |  |
| --- | --- |
| 函数名 | PX\_VectorFree(x) |
| 说明 | **释放容器,同时释放内存,若需要重新使用该容器必须对其重新初始化** |
| 参数 | x容器指针 |
| 返回值 | - |

##### 拷贝容器

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_VectorCopy(px\_vector \*destvec,px\_vector \*resvec); |
| 说明 | **拷贝容器** |
| 参数 | destvec 目标容器  resvec 原容器 |
| 返回值 | 成功返回PX\_TRUE,否者PX\_FALSE |

##### 重置容器的大小

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_VectorResize(px\_vector \*vec,px\_int size); |
| 说明 | **重置容器的大小** |
| 参数 | vec目标容器  size重置大小  注意,重置大小的容器数据将直接丢失 |
| 返回值 | 成功返回PX\_TRUE,否者PX\_FALSE |

#### 字符串映射表

##### 初始化映射表

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MapInitialize(px\_memorypool \*mp,px\_map \*hashmap); |
| 说明 | **初始化一个**映射**表** |
| 参数 | mp,内存池  hashmap 映射表 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

##### 插入元素

|  |  |
| --- | --- |
| 函数名 | PX\_HASHMAP\_RETURN PX\_MapPut (px\_map \* m, px\_char\* key, px\_void \* value); |
| 说明 | **插入一个节点到表中** |
| 参数 | m 哈希表  key 映射字符串  value 值 |
| 返回值 | 如果成功返回PX\_HASHMAP\_RETURN\_OK |

|  |  |
| --- | --- |
| 函数名 | PX\_HASHMAP\_RETURN PX\_MapPutInt (px\_map \* m, px\_char\* key, px\_int value); |
| 说明 | **插入一个int节点到表中** |
| 参数 | m 哈希表  key 映射字符串  value 值 |
| 返回值 | 如果成功返回PX\_HASHMAP\_RETURN\_OK |

|  |  |
| --- | --- |
| 函数名 | PX\_HASHMAP\_RETURN PX\_MapPutFloat (px\_map \* m, px\_char\* key, px\_int value); |
| 说明 | **插入一个float节点到表中** |
| 参数 | m 哈希表  key 映射字符串  value 值 |
| 返回值 | 如果成功返回PX\_HASHMAP\_RETURN\_OK |

##### 查找元素

|  |  |
| --- | --- |
| 函数名 | px\_void \*PX\_MapGet(px\_map \* m, px\_char\* stringkey); |
| 说明 | **查找哈希表对应key的元素** |
| 参数 | m 哈希表  key 映射字符串 |
| 返回值 | 对于返回类型 |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MapGetInt(px\_map \* m, px\_char\* stringkey,px\_int \*v); |
| 说明 | **查找哈希表对应key的int元素** |
| 参数 | m 哈希表  key 映射字符串  v返回赋值指针 |
| 返回值 | 如果成功返回PX\_TRUE,否则返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_MapGetFloat(px\_map \* m, px\_char\* stringkey); |
| 说明 | **查找哈希表对应key的float元素** |
| 参数 | m 哈希表  key 映射字符串  v返回赋值指针 |
| 返回值 | 如果成功返回PX\_TRUE,否则返回PX\_FALSE |

##### 删除元素

|  |  |
| --- | --- |
| 函数名 | PX\_HASHMAP\_RETURN PX\_MapErase (px\_map \* m, px\_char\* key); |
| 说明 | **删除对应key的元素** |
| 参数 | m 哈希表  key 映射字符串 |
| 返回值 | 如果成功返回PX\_HASHMAP\_RETURN\_OK |

##### 释放map

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MapFree(px\_map \* m) |
| 说明 | **删除对应map所有元素并释放内存资源** |
| 参数 | m 哈希表 |
| 返回值 | - |

#### 自适应内存

PainterEngine Core提供自动内存类型px\_memory,注意,px\_ memory使用前,都应该使用PX\_MemoryInit进行初始化,并在使用完毕后使用PX\_MemoryFree释放内存

##### 初始化

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MemoryInitialize(px\_memorypool \*mp,px\_memory \*memory); |
| 说明 | **对自适应内存结构进行初始化** |
| 参数 | mp 容器部署内存池指针  memory 内存指针 |

##### 拼接内存

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MemonyCat(px\_memory \*memory,px\_void \*buffer,px\_int size); |
| 说明 | **对自适应内存进行数据拼接** |
| 参数 | memory内存指针  buffer 需要拼接的内存指针  size 拼接大小 |
| 返回值 | 若成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 重置内存分配大小

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MemonyResize(px\_memory \*memory, px\_int size); |
| 说明 | **重置内存分配大小** |
| 参数 | memory内存指针  size 新的分配大小 |
| 返回值 | 若成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 释放内存

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MemoryFree(px\_memory \*memory); |
| 说明 | **释放内存** |
| 参数 | memory 内存指针 |

##### 取内存数据

|  |  |
| --- | --- |
| 函数名 | px\_byte \* PX\_MemoryData(px\_memory \*memory); |
| 说明 | **取内存数据指针** |
| 参数 | memory 内存指针 |
| 返回值 | 内存数据指针 |

##### 内存查找

|  |  |
| --- | --- |
| 函数名 | px\_byte \* PX\_MemoryFind(px\_memory \*memory,px\_void \*buffer,px\_int size); |
| 说明 | **在内存中查找匹配内存** |
| 参数 | memory 内存指针  buffer 匹配内存缓存  size 匹配内存缓存大小 |
| 返回值 | 如果找到返回该内存数据指针,否者返回PX\_NULL |

##### 内存拷贝

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MemoryCopy(px\_memory \*memory,px\_void \*buffer,px\_uint startoffset,px\_int size); |
| 说明 | **拷贝内存到目标内存结构中,注意,这个函数会依据拷贝内存大小自动调整内存部署** |
| 参数 | memory 内存指针  buffer 拷贝内存缓存  startoffset 拷贝到的起始地址  size 拷贝内存缓存大小 |
| 返回值 |  |

##### 内存清理

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MemoryClear(px\_memory \*memory); |
| 说明 | **清空内存,这个函数并不会释放占用内存空间** |
| 参数 | memory 内存指针 |
| 返回值 |  |

##### 移除内存

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MemoryRemove(px\_memory \*memory,px\_int start,px\_int end) |
| 说明 | **移除一块内存区域,在区域之后的内存数据将自动拼接到移除的位置中** |
| 参数 | memory 内存指针  start 开始索引  end 结束索引  \*开始索引必须小于等于结束索引,结束索引不得大于等于实际已用内存大小 |
| 返回值 |  |

#### 内存自适应字符串

PainterEngine Core提供自动内存调节的字符串类型px\_string,注意,px\_string的需要更多的性能开销,字符串使用前,都应该使用PX\_StringInit进行初始化,并在使用完毕后使用PX\_StringFree释放内存

##### 初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_StringInitialize(px\_memorypool \*mp,px\_string \*str); |
| 说明 | **对字符串结构进行初始化** |
| 参数 | mp 容器部署内存池指针  str 字符串指针 |
| 返回值 | 成功返回PX\_TRUE,否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringInitFromConst(px\_string \*str,px\_char \*conststr); |
| 说明 | **使用一个字符串常量对字符串结构初始化,这个字符串额外占据的内存为0** |
| 参数 | str 字符串指针  conststr 字符串常量指针 |

##### 转换

###### 转换为整数

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_StringToInteger(px\_string \*str); |
| 说明 | **转换字符串为整数,支持十六进制转换(以0x或0X开头)** |
| 参数 | str 字符串指针 |
| 返回值 | 转换整数结果, |

###### 转换为小数

|  |  |
| --- | --- |
| 函数名 | px\_float PX\_StringToFloat(px\_string \*str); |
| 说明 | **转换字符串为浮点数** |
| 参数 | str 字符串指针 |
| 返回值 | 转换浮点结果, |

##### 设置字符串文本

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_StringSet(px\_string \*str,px\_char fmt[]); |
| 说明 | **设置字符串文本** |
| 参数 | str 字符串指针  fmt 需要设置的文本 |
| 返回值 | 成功返回PX\_TRUE,否者PX\_FALSE(内存不足) |

##### 去除首位空格

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringTrim(px\_string \*str); |
| 说明 | **去除首位空格,** |
| 参数 | str 字符串指针 |
| 返回值 | - |

##### 取字符串长度

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_StringLen(px\_string \*str); |
| 说明 | **取得字符串长度** |
| 参数 | str 字符串指针 |
| 返回值 | 字符长度 |

##### 字符串拼接

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringCatChar(px\_string \*str,px\_char ch); |
| 说明 | 将ch拼接到字符串str的尾部 |
| 参数 | str 字符串指针  ch 拼接字母 |
| 返回值 | - |
| 函数名 | px\_void PX\_StringCat(px\_string \*str,px\_char \*str2); |
| 说明 | **将str2拼接到字符串str的尾部** |
| 参数 | str 字符串指针  str2 拼接字符串指针 |
| 返回值 | - |

##### 释放字符串

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringFree(px\_string \*str); |
| 说明 | **释放字符串内存** |
| 参数 | str 字符串指针 |

##### 清理字符串

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringClear(px\_string \*str); |
| 说明 | **清理字符串文本(不释放内存)** |
| 参数 | str 字符串指针 |

##### 插入字符

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_StringInsertChar(px\_string \*str,px\_int index,px\_char ch); |
| 说明 | **将一个字符插入到字符串中** |
| 参数 | str 字符串指针  index 插入的位置索引  ch 要插入的字符 |

##### 删除字符

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_StringRemoveChar(px\_string \*str,px\_int index); |
| 说明 | **将一个字符到从字符串中删除** |
| 参数 | str 字符串指针  index 插入的位置索引  ch 要插入的字符 |

##### 裁剪字符串

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringTrimLeft(px\_string \*str,px\_int leftCount); |
| 说明 | **裁剪字符串左边的字符** |
| 参数 | str 字符串指针  leftCount裁去左边的字符数 |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringTrimRight(px\_string \*str,px\_int RightCount); |
| 说明 | **裁剪字符串右边的字符** |
| 参数 | str 字符串指针  RightCount裁去左边的字符数 |

##### 格式化字符串

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_StringFormat8(px\_string \*str,px\_char fmt[],px\_stringformat \_1, px\_stringformat \_2, px\_stringformat \_3, px\_stringformat \_4,px\_stringformat \_5, px\_stringformat \_6, px\_stringformat \_7, px\_stringformat \_8);  px\_bool PX\_StringFormat7(px\_string \*str,px\_char fmt[],px\_stringformat \_1, px\_stringformat \_2, px\_stringformat \_3, px\_stringformat \_4,px\_stringformat \_5, px\_stringformat \_6, px\_stringformat \_7);  px\_bool PX\_StringFormat6(px\_string \*str,px\_char fmt[],px\_stringformat \_1, px\_stringformat \_2, px\_stringformat \_3, px\_stringformat \_4,px\_stringformat \_5, px\_stringformat \_6);  px\_bool PX\_StringFormat5(px\_string \*str,px\_char fmt[],px\_stringformat \_1, px\_stringformat \_2, px\_stringformat \_3, px\_stringformat \_4,px\_stringformat \_5);  px\_bool PX\_StringFormat4(px\_string \*str,px\_char fmt[],px\_stringformat \_1, px\_stringformat \_2, px\_stringformat \_3, px\_stringformat \_4);  px\_bool PX\_StringFormat3(px\_string \*str,px\_char fmt[],px\_stringformat \_1, px\_stringformat \_2, px\_stringformat \_3);  px\_bool PX\_StringFormat2(px\_string \*str,px\_char fmt[],px\_stringformat \_1, px\_stringformat \_2);  px\_bool PX\_StringFormat1(px\_string \*str,px\_char fmt[],px\_stringformat \_1); |
| 说明 | **格式化字符串** |
| 参数 | 参考px\_sprintf1…的实现 |
|  |  |

##### 筛选

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_StringIsNumeric(px\_string \*str); |
| 说明 | **判断字符串是否为合法数字** |
| 参数 | str 字符串指针 |
| 返回值 | 如果是返回PX\_TRUE,否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_StringIsFloat(px\_string \*str); |
| 说明 | **判断字符串是否为合法浮点数** |
| 参数 | str 字符串指针 |
| 返回值 | 如果是返回PX\_TRUE,否者PX\_FALSE |

##### 字符串替换1

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringReplaceRange(px\_string \*str,px\_int startindex,px\_int endindex,px\_char \*replaceto) |
| 说明 | **字符串替换,将目标范围的字符串替换为replaceto里的字符串** |
| 参数 | str 字符串指针  startindex 开始索引  endindex 结束索引  replaceto 替换成的字符串 |
| 返回值 | - |

##### 字符串替换2

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_StringReplace(px\_string \*str,px\_char \*source,px\_char \*replaceto); |
| 说明 | **字符串替换,将str中包含source的文本都替换为replaceto** |
| 参数 | str 字符串指针  source 需要搜索的原始文本  replaceto 替换成的文本 |
| 返回值 | - |

### 几何绘制

#### 线

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawLine(px\_surface \*psurface, px\_int x0, px\_int y0, px\_int x1, px\_int y1 ,pt\_int lineWidth, px\_color color); |
| 说明 | **绘制一个反走样线段** |
| 参数 | psurface 渲染表面  x0 y0起始点坐标  x1 y1 终点坐标  lineWidth 线宽  color 颜色 |

#### 边框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawBorder(px\_surface \*psurface, px\_int left, px\_int top, px\_int rignt, px\_int bottom ,px\_int lineWidth,px\_color color); |
| 说明 | **绘制一个边框** |
| 参数 | psurface 渲染表面  left top right bottom 位置描述  lineWidth 边框宽度像素  color 颜色 |

#### 矩形

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawRect(px\_surface \*psurface, px\_int left, px\_int top, px\_int right, px\_int bottom,px\_color color); |
| 说明 | **绘制一个实心矩形** |
| 参数 | psurface 渲染表面  left top right bottom 位置描述  color 颜色 |

#### 实心圆

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawSolidCircle(px\_surface \*psurface, px\_int x,px\_int y,px\_int Radius,px\_color color ); |
| 说明 | **绘制一个反走样实心圆** |
| 参数 | psurface 渲染表面  x，y 圆心  radius 半径  color 颜色 |

#### 圆

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawCircle(px\_surface \*psurface,px\_int x,px\_int y,px\_int Radius ,pt\_int lineWidth,px\_color color); |
| 说明 | **绘制一个反走样圆** |
| 参数 | psurface 渲染表面  x，y 圆心  radius 半径  lineWidth 线宽  color 颜色 |

#### 圆笔

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawPenCircle(px\_surface \*psurface, px\_float x,px\_float y,px\_float Radius,px\_color color) |
| 说明 | **绘制一个反走样圆,这个算法用于高质量圆笔的绘制,可用这个算法组合为线段路径** |
| 参数 | psurface 渲染表面  x，y 圆心  radius 半径  lineWidth 线宽  color 颜色 |

#### 环

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawRing(px\_surface \*psurface, px\_int x,px\_int y,px\_int Radius,px\_int lineWidth,px\_color color,px\_uint start\_angle,px\_uint end\_angle) |
| 说明 | **绘制一个反走样环** |
| 参数 | psurface 渲染表面  x，y 环心  radius 半径  lineWidth 线宽  color 颜色  start\_angle 起始角度  end\_angle 终止角度  这个环遵循顺时针方向 |

#### 扇形

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawSector(px\_surface \*psurface, px\_int x,px\_int y,px\_int Radius\_outside,px\_int Radius\_inside,px\_color color,px\_uint start\_angle,px\_uint end\_angle); |
| 说明 | **绘制一个反走样扇形** |
| 参数 | psurface 渲染表面  x，y 环心  radius 半径  outside 外径  inside 内径  color 颜色  start\_angle 起始角度  end\_angle 终止角度  这个扇形遵循顺时针方向 |

#### 圆角矩形

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawRoundRect(px\_surface \*psurface, px\_int left, px\_int top, px\_int right, px\_int bottom,px\_float roundRaduis,px\_float linewidth,px\_color color); |
| 说明 | **绘制一个反走样圆角矩形** |
| 参数 | psurface 渲染表面  left top right bottom 位置描述  roundRadius 圆角半径  lineWidth 线宽  color 颜色 |

#### 实心圆角矩形

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawSolidRoundRect(px\_surface \*psurface, px\_int left, px\_int top, px\_int right, px\_int bottom,px\_float roundRaduis,px\_color color); |
| 说明 | **绘制一个反走样实心圆角矩形** |
| 参数 | psurface 渲染表面  left top right bottom 位置描述  roundRadius 圆角半径  color 颜色 |

#### 描边路径

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawPath(px\_surface \*psurface, px\_point path[],px\_int pathCount,px\_float linewidth,px\_color color) |
| 说明 | **描边路径**  **\*注意:渲染算法使用圆点插值算法进行绘制以获得高质量的描线,alpha值因为step也会有所放大(约4倍),这个算法耗费性能资源较大,这个渲染算法应该谨慎用于实时渲染表面** |
| 参数 | psurface 渲染表面  path 路径上所有的点  pathCount 点的数量  linewidth 线宽  color 颜色 |
|  |  |

#### 三角形

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawTriangle(px\_surface \*psurface,px\_point2D p0,px\_point2D p1,px\_point2D p2,px\_color color); |
| 说明 | **绘制一个反走样三角形** |
| 参数 | psurface 渲染表面  p0 p1 p2 三角形三点  color 颜色 |
|  |  |

#### 箭头

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawArrow(px\_surface \*psurface,px\_point2D p0,px\_point2D p1,px\_float size,px\_color color) |
| 说明 | **绘制一个由p0指向p1的箭头** |
| 参数 | psurface 渲染表面  p0 p1 箭头向量两点  size 箭头尺寸  color 颜色 |
|  |  |

#### 绘制贝塞尔曲线

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_GeoDrawBezierCurvePoint(px\_surface \*rendersurface,px\_point pt[],px\_int pt\_count,px\_float t,px\_float radius,px\_color clr); |
| 说明 | **绘制一条圆笔控制步长的由多个控制点组成的贝塞尔曲线** |
| 参数 | Rendersurface 渲染表面  pt 贝塞尔控制点数组  pt\_count 控制点个数  t 圆笔步长  radius 圆笔半径  clr 圆笔每点颜色 |
|  |  |

### 图形图像

#### 渲染表面

##### 颜色

|  |  |
| --- | --- |
| 函数名 | px\_color PX\_COLOR(px\_uchar a,px\_uchar r,px\_uchar g,px\_uchar b); |
| 说明 | **构造颜色** |
| 参数 | A,r,g,b颜色分量 |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ColorIncrease(px\_color \*color,px\_uchar inc); |
| 说明 | **增量颜色值** |
| 参数 | Color 指向需要增量的颜色结构  Inc 增量值 |

|  |  |
| --- | --- |
| 函数名 | px\_color PX\_ColorAdd(px\_color color1,px\_color color2); |
| 说明 | **颜色相加** |
| 参数 | Color1，color2 需要相加的两个颜色 |
| 返回值 | 返回相加的颜色结果 |

|  |  |
| --- | --- |
| 函数名 | px\_color PX\_ColorSub(px\_color color1,px\_color color2); |
| 说明 | **颜色相减** |
| 参数 | Color1，color2 需要相减的两个颜色 |
| 返回值 | 返回相减的颜色结果 |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ColorEqual(px\_color color1,px\_color color2); |
| 说明 | **判断两个颜色是否相等** |
| 参数 | Color1，color2 需要判断的两个颜色 |
| 返回值 | 如果相等返回PX\_TRUE,否者返回PX\_FALSE |

##### 表面操作

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SurfaceCreate(px\_memorypool \*mp,px\_uint height,px\_uint width,px\_surface \*surface); |
| 说明 | **创建一个渲染表面,它是用于描述渲染内存的结构体，仅支持32位BGRA色彩格式。** |
| 参数 | Mp 内存池指针  Height 高  Width 宽  Surface 渲染表面指针 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SurfaceFree(px\_surface \*psurface); |
| 说明 | **释放一个渲染表面** |
| 参数 | psurface 指向需要释放的渲染表面 |

##### 像素操作

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SurfaceDrawPixel(px\_surface \*ps,px\_int x,px\_int y,px\_color color); |
| 说明 | **绘制一个像素** |
| 参数 | ps 指向表面  x，y 坐标  color 颜色 |
| 返回值 |  |

|  |  |
| --- | --- |
| 函数名 | PX\_SUFACECOLOR(Surface,X,Y) |
| 说明 | **查找像素颜色** |
| 参数 | Surface指向表面  X，Y 坐标 |
| 返回值 | 对应px\_color |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SetPixel(px\_surface \*ps,px\_int x,px\_int y,px\_color color); |
| 说明 | **设置一个像素** |
| 参数 | ps 指向表面  x，y 坐标  color 颜色 |
| 返回值 |  |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SurfaceSetRect(px\_surface \*psurface, px\_int left, px\_int top, px\_int right, px\_int bottom,px\_color color) |
| 说明 | **将一个渲染表面的一个矩形区域设置为某种颜色** |
| 参数 | psurface 渲染表面  left top right bottom 位置描述  color 颜色 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_SurfaceMemorySize(px\_uint width,px\_uint height); |
| 说明 | **预估一个表面的占用内存(实际占用依据内存池结构应适当增大)** |
| 参数 | width 表面宽度  height 表面高度 |
| 返回值 | - |

##### 设置渲染限制区域

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SurfaceSetLimit(px\_surface \*ps,px\_int limit\_left,px\_int limit\_top,px\_int limit\_right,px\_int limit\_bottom); |
| 说明 | **设置渲染的限制区域,该区域不得超出渲染表面的区域(否者函数会对其进行裁剪)**  **这个限制区域将会PX\_SurfaceDrawPixel和PX\_SurfaceSetPixel生效** |
| 参数 | ps  limit\_left 限制区域左上角x坐标  limit\_top 限制区域左上角y坐标  limit\_right 限制区域右下角x坐标  limit\_bottom 限制区域右下角y坐标 |
| 返回值 | - |

##### 取得当前渲染限制信息

|  |  |
| --- | --- |
| 函数名 | PX\_SurfaceLimitInfo PX\_SurfaceGetLimit(px\_surface \*ps); |
| 说明 | **取得渲染的限制区域** |
| 参数 | ps 渲染表面 |
| 返回值 | -限制信息 |

##### 设置当前渲染限制信息

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SurfaceSetLimitInfo(px\_surface \*ps,PX\_SurfaceLimitInfo info); |
| 说明 | **设置渲染的限制区域** |
| 参数 | ps 渲染表面  info 限制信息 |
| 返回值 |  |

#### 纹理

px\_texture是PainterEngine的纹理对象结构，建议所有的图像数据都最终加载为px\_texture

其本质上仍然是一个surface,其指针可以和px\_surface互换使用.

##### 创建纹理

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TextureCreate(px\_memorypool \*mp,px\_texture \*tex,px\_int width,px\_int height); |
| 说明 | **创建一个Texture表面** |
| 参数 | mp 内存池指针  tex 输出表面指针  width 宽  height 高 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TextureCreateFromMemory(px\_memorypool \*mp,px\_void \*data,px\_int size,px\_texture \*tex); |
| 说明 | **从支持的数据创建一个Texture表面（如bmp或TRaw）** |
| 参数 | mp 内存池指针  data 数据指针  size 宽  tex 纹理格式 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

##### 缩放纹理

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TextureCreateScale(px\_memorypool \*mp,px\_texture \*resTexture,px\_int newWidth,px\_int newHeight,px\_texture \*out); |
| 说明 | **使用窗采样缩放一个纹理并创造缩放后的纹理** |
| 参数 | mp 内存池指针  resTexture 原纹理指针  newWidth newHeight 缩放后的大小  out 缩放后的纹理指针,在使用完毕后改指针必须被释放 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

##### 拷贝纹理

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TextureCopy(px\_memorypool \*mp,px\_texture \*resTexture, px\_texture \*out); |
| 说明 | **深拷贝一个纹理到目标纹理中** |
| 参数 | mp 内存池指针  resTexture 原纹理指针  out 拷贝到目标纹理 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

##### 渲染纹理

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRender(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend); |
| 说明 | **渲染一个纹理到表面** |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND; |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRenderPixelShader(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TexturePixelShader shader,px\_void \*ptr); |
| 说明 | **渲染一个纹理到表面,每个像素渲染将会调用**shader |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  shader pixels shader函数  ptr 传递给PixelsShader的指针 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRenderRotation(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend,px\_int Angle); |
| 说明 | **渲染一个纹理到表面,并顺时针旋转一个角度** |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND;  Angle 旋转的角度  注意该函数使用双线性插值滤波运算,对于性能要求较高的渲染函数可以使用PX\_TextureRenderEx,以较低的显示损失获得8倍左右的性能 |
| 返回值 | - |
| 函数名 | px\_void PX\_TextureRenderRotation\_vector(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend,px\_point p\_vector); |
| 说明 | 渲染一个纹理到表面,并顺时针旋转一个角度 |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND;  P\_vector 旋转向量,以x轴正方向为基准  注意该函数使用双线性插值滤波运算,对于性能要求较高的渲染函数可以使用PX\_TextureRenderEx,以较低的显示损失获得8倍左右的性能 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRenderRotation\_sincos(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend,px\_float sin,px\_float cos); |
| 说明 | 渲染一个纹理到表面,并顺时针旋转一个角度 |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND;  Sin cos 旋转矩阵对于sin cos值  注意该函数使用双线性插值滤波运算,对于性能要求较高的渲染函数可以使用PX\_TextureRenderEx,以较低的显示损失获得8倍左右的性能 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRenderEx(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend,px\_float scale,px\_float rotation); |
| 说明 | **渲染一个纹理到表面,注意,这个渲染方式的速度不如PX\_TextureRender,但提供了纹理的缩放和旋转(同样处于效率考虑,其缩放旋转均采用单点采样,其质量不如使用纹理缩放的函数,但速度快于这两函数,对于高精度旋转和缩放动画,采用该函数可以获得一个折中的效率,内存空间与动画效果)** |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND;  scale 缩放大小  rotation 旋转角度 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRenderEx\_sincos(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend,px\_float scale,px\_float sin,px\_float cos); |
| 说明 | **渲染一个纹理到表面,注意,这个渲染方式的速度不如PX\_TextureRender,但提供了纹理的缩放和旋转(以旋转矩阵sin cos形式,同样处于效率考虑,其缩放旋转均采用单点采样,其质量不如使用纹理缩放的函数,但速度快于这两函数,对于高精度旋转和缩放动画,采用该函数可以获得一个折中的效率,内存空间与动画效果)** |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND;  scale 缩放大小  sin 旋转矩阵sin值  cos 旋转矩阵 cos值 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRenderEx(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend,px\_float scale,px\_float rotation); |
| 说明 | **渲染一个纹理到表面,注意,这个渲染方式的速度不如PX\_TextureRender,但提供了纹理的缩放和旋转(同样处于效率考虑,其缩放旋转均采用单点采样,其质量不如使用纹理缩放的函数,但速度快于这两函数,对于高精度旋转和缩放动画,采用该函数可以获得一个折中的效率,内存空间与动画效果)** |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND;  scale 缩放大小  rotation 旋转角度 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRenderEx\_vector(px\_surface \*psurface,px\_texture \*tex,px\_int x,px\_int y ,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend,px\_float scale,px\_point p\_vector); |
| 说明 | **渲染一个纹理到表面,注意,这个渲染方式的速度不如PX\_TextureRender,但提供了纹理的缩放和旋转(以旋转矩阵vector形式,同样处于效率考虑,其缩放旋转均采用单点采样,其质量不如使用纹理缩放的函数,但速度快于这两函数,对于高精度旋转和缩放动画,采用该函数可以获得一个折中的效率,内存空间与动画效果)**  **默认的指向方向为** |
| 参数 | psurface 渲染到的表面  px\_texture需要渲染的纹理  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND;  scale 缩放大小  指向方向 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureRegionRender(px\_surface \*psurface,px\_texture \*resTexture,px\_int x,px\_int y,px\_int oft\_left,px\_int oft\_top,px\_int oft\_right,px\_int oft\_bottom,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend) |
| 说明 | **渲染部分纹理** |
| 参数 | psurface 渲染表面  ptexture 纹理  x,y 绘制参考点  left,right,top,bottom 需要绘制的纹理位置描述  refPoint 参考点类型的位置描述  blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  typedef struct  {  float hdr\_R; //HDR of Red  float hdr\_G; //HDR of Green  float hdr\_B; //HDR of Blue  float alpha; //Blend of alpha  }PX\_TEXTURERENDER\_BLEND; |
| 返回值 | - |

##### 创建旋转纹理

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TextureCreateRotationAngle(px\_memorypool \*mp,px\_texture \*resTexture,px\_float Angle,px\_texture \*out); |
| 说明 | **将一个纹理按角度进行顺时针旋转并创建一个新纹理** |
| 参数 | mp 内存池指针  resTexture 原纹理  Angle 旋转的角度  out 输出纹理 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TextureRotationAngleToTexture(px\_texture \*resTexture,px\_float Angle,px\_texture \*out); |
| 说明 | **将一个纹理按角度进行顺时针旋转并拷贝到目标纹理** |
| 参数 | resTexture 原纹理  Angle 旋转的角度  out 输出纹理 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TextureCreateRotationRadian(px\_memorypool \*mp,px\_texture \*resTexture,px\_float Rad,px\_texture \*out); |
| 说明 | **将一个纹理按弧度进行顺时针旋转并创建一个新纹理** |
| 参数 | mp 内存池指针  resTexture 原纹理  Rad旋转的弧度  out 输出纹理 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

##### 释放纹理

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureFree(px\_texture \*tex); |
| 说明 | **释放纹理内存** |
| 参数 | tex 纹理指针 |
| 返回值 | - |

##### 填充纹理

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureFill(px\_memorypool \*mp,px\_texture \*ptexture,px\_int x,px\_int y,px\_color test\_color,px\_color fill\_color) |
| 说明 | **填充纹理** |
| 参数 | mp 运行计算内存池  ptexture 纹理  x,y填充位置  test\_color 测试颜色  fill\_color 填充颜色 |
| 返回值 | - |

##### 获取纹理可见数据包围盒

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TextureGetVisibleRange(px\_texture \*ptexture,px\_int \*pLeft,px\_int \*pRight,px\_int \*pTop,px\_int \*pBottom); |
| 说明 | **填充纹理** |
| 参数 | ptexture 纹理  pleft,pright,ptop,pbottom 输出可见包围信息 |
| 返回值 | - |

#### 图像算子

##### Priwitt算子

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_PriwittX(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Priwitt算子 水平方向 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_PriwittY(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Priwitt算子 垂直方向 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_Priwitt(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Priwitt算子 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

##### Sobel算子

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_SobelX(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 水平方向 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_SobelY(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 垂直方向 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_Sobel (px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

##### Roberts算子

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_RobertsX(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 水平方向 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_RobertsY(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 垂直方向 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_Roberts (px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

##### Laplacian算子

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_LaplacianX(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 水平方向 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_LaplacianY(px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 垂直方向 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ImageFilter\_Laplacian (px\_texture \*ptexture,px\_float out[]); |
| 说明 | Sobel算子 |
| 参数 | ptexture 纹理指针  out 输出矩阵，大小应该至少是纹理长宽 |
| 返回值 | - |

#### 轮廓

PX\_Shape是painterEngine内建的轮廓格式,轮廓可以被认为是仅带有灰度及位置信息的纹理,轮廓的每个位置信息都由一个256阶的灰度表示(1字节)在渲染轮廓的时候,需要制定一个颜色信息以对轮廓进行渲染.

##### 创建轮廓

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ShapeCreate(px\_memorypool \*mp,px\_shape \*shape,px\_int width,px\_int height); |
| 说明 | **创建一个轮廓** |
| 参数 | mp 内存池指针  shape 输出轮廓  width 宽  height 高 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

##### 创建纹理轮廓

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ShapeCreateFromTexture(px\_memorypool \*mp,px\_shape \*shape,px\_texture \*texture); |
| 说明 | **从纹理创建一个轮廓** |
| 参数 | mp 内存池指针  shape 输出轮廓  texture 映射纹理 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ShapeCreateFromMemory(px\_memorypool \*mp,px\_void \*data,px\_int size,px\_shape \*shape); |
| 说明 | **从内存创建一个轮廓(支持TRaw内建格式)** |
| 参数 | mp 内存池指针  shape 输出轮廓  data 数据  size 数据大小 |
| 返回值 | 若成功返回PX\_TRUE 否者PX\_FALSE |

##### 渲染轮廓

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ShapeRender(px\_surface \*psurface,px\_shape \*shape,px\_int x,px\_int y,PX\_ALIGN refPoint,px\_color blendColor); |
| 说明 | **渲染一个轮廓到表面** |
| 参数 | psurface 渲染到的表面  px\_shape需要渲染的轮廓  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  blendcolor 渲染颜色 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ShapeRenderEx(px\_surface \*psurface,px\_shape \*shape,px\_int x,px\_int y,PX\_ALIGN refPoint,px\_color blendColor,px\_float scale,px\_float Angle); |
| 说明 | **旋转缩放后渲染一个轮廓到表面** |
| 参数 | psurface 渲染到的表面  shape 需要渲染的轮廓  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  blendcolor 渲染颜色  scale 缩放大小  rotation 旋转角度 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ShapeRenderEx\_sincos(px\_surface \*psurface,px\_shape \*shape,px\_int x,px\_int y,PX\_ALIGN refPoint,px\_color blendColor,px\_float scale,px\_float sinx,px\_float cosx); |
| 说明 | **旋转后渲染一个轮廓到表面** |
| 参数 | psurface 渲染到的表面  shape 需要渲染的轮廓  x，y偏移量(该坐标以纹理左上角为参照)  refPoint 参考中心点，用于表示纹理绘制的相对位置  blend blend类型结构体，用于调整绘制纹理的alpha，hdr值  当它为PX\_NULL时，表示采用默认blend值  blendcolor 渲染颜色  sinx cosx,旋转矩阵对应sinx cosx位置值,这个函数在以向量为标准的旋转标定渲染中尤为有用 |
| 返回值 | - |

##### 释放轮廓

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ShapeFree(px\_shape \*shape); |
| 说明 | **释放轮廓** |
| 参数 | shape 轮廓指针 |
| 返回值 | - |

#### 位图

##### 验证位图数据合法性

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_BitmapVerify(void \*BitmapBuffer,px\_int size); |
| 说明 | **验证数据是否为合法位图数据** |
| 参数 | BitmapBuffer 位图数据  Size 数据长度 |
| 返回值 | 如果合法返回PX\_TRUE,否者PX\_FALSE |
|  |  |

##### 取得位图高度

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_BitmapGetHeight(void \*BitmapBuffer); |
| 说明 | **取得位图高度** |
| 参数 | BitmapBuffer 位图数据 |
| 返回值 | 位图的高度(可能为负数) |
|  |  |

##### 取得位图宽度

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_BitmapGetWidth(void \*BitmapBuffer); |
| 说明 | **取得位图宽度** |
| 参数 | BitmapBuffer 位图数据 |
| 返回值 | 位图的宽度(可能为负数) |
|  |  |

##### 取得位图像素数据

|  |  |
| --- | --- |
| 函数名 | px\_word PX\_BitmapGetBitCount(void \*BitmapBuffer); |
| 说明 | **取得位图像素数据** |
| 参数 | BitmapBuffer 位图数据 |
| 返回值 | 位图的位图像素数据 |

##### 渲染位图到表面

|  |  |
| --- | --- |
| 函数名 | void PX\_BitmapRender(px\_surface \*psurface,void \*BitmapBuffer,px\_int BufferSize,int x,int y); |
| 说明 | **渲染位图到表面** |
| 参数 | psurface渲染表面  BitmapBuffer 位图数据  BufferSize 数据长度  x,y偏移量 |
| 返回值 |  |

##### Surface转位图数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_BitmapBuild(px\_surface \*psurface,px\_char \*BitmapBuffer,px\_int \*size); |
| 说明 | **将一个px\_surface转换成位图格式数据** |
| 参数 | psurface渲染表面  BitmapBuffer 输出数组,如果为PX\_NULL,表示仅计算需要的大小  size 实际输出的字节大小 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### TRaw纹理格式

typedef struct \_PX\_TRaw\_Header

{

px\_dword Magic;//TRAW 0x57415254

px\_int Width;

px\_int Height;

}PX\_TRaw\_Header;

TRaw文件头，TRaw文件是一个后缀为tex的文件

##### 判断是否为TRaw合法文件格式

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TRawVerify(px\_void \*data,px\_int size); |
| 说明 | **判断是否为TRaw合法文件格式** |
| 参数 | data 数据指针 size 数据大小 |
| 返回值 | 返回TRUE表示合法数据否者非合法数据 |

##### 取得TRaw文件的图像宽度

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_TRawGetWidth(px\_void \*data); |
| 说明 | **取得TRaw文件的图像宽度** |
| 参数 | data 数据指针 |
| 返回值 | 返回图像宽度 |

##### 取得TRaw文件的图像高度

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_TRawGetHeight(px\_void \*data); |
| 说明 | **取得TRaw文件的图像高度** |
| 参数 | data 数据指针 |
| 返回值 | 返回图像宽度 |

##### 绘制TRaw数据

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TRawRender(px\_surface \*psurface,px\_void \*data,px\_int x,px\_int y); |
| 说明 | **绘制TRaw数据** |
| 参数 | psurface 表面指针  data TRaw数据  x，y偏移量 |
| 返回值 | - |

##### 将pSurface转换为TRAW格式数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TRawBuild(px\_surface \*psurface,px\_byte \*TRawBuffer,px\_int \*size); |
| 说明 | **将pSurface转换为TRAW格式数据** |
| 参数 | psurface 表面指针  TRawBuffer 输出数据(当TRAWBuffer不为PX\_NULL时)  size 输出大小 |
| 返回值 | - |

#### Delaunary三角划分

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_DelaunaryPointsBuild(px\_memorypool \*mp,px\_point2D pt[],px\_int count,px\_vector \*out\_triangles); |
| 说明 | **将点集以Delaunary Triangle进行空间划分,生成图形是一个凸包** |
| 参数 | mp 运行计算内存池  pt 点集  count 点集点数量  out\_Triangles 输出三角形 |
| 返回值 | 成功返回PX\_TRUE,否者PX\_FALSE |

#### 带限制区域的Delaunary三角建立

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_PointsMeshBuild(px\_memorypool \*mp,px\_point2D limit\_line\_pt[],px\_int line\_pt\_count,px\_point2D pt[],px\_int pt\_count,px\_vector \*out\_triangles,PX\_DELAUNAY\_RETURN\_TYPE type); |
| 说明 | **将点集以Delaunary规则建立 Triangle，同时，剔除limit\_line\_pt绘制的闭环之外的三角形** |
| 参数 | mp 运行计算内存池  limit\_line\_pt 用于绘制封闭图形的线段集  line\_pt\_count用于绘制封闭图形的线段集点个数  pt 点集  count 点集点数量  out\_Triangles 输出三角形  type 返回数据类型(以三角坐标返回或以索引形式返回) |
| 返回值 | 成功返回PX\_TRUE,否者PX\_FALSE |

#### 纹理特效

##### 发光

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_EffectShine(px\_surface \*s,px\_int radius,px\_color color,px\_float intension); |
| 说明 | **对一个纹理施加发光效果** |
| 参数 | S 目标纹理  Radius 半径  Color 颜色  Intension 强度 |
| 返回值 | - |

##### 描边

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_EffectOutline(px\_surface \*s,px\_int radius,px\_color color); |
| 说明 | **对一个纹理施加描边效果** |
| 参数 | S 目标纹理  Radius 半径  Color 颜色 |
| 返回值 | - |

### 字体

PainterEngine 内嵌了默认ANSI字模库,同时运行用户加载自己的字模库进行字体绘制

#### 绘制字体

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_FontModuleDrawText(px\_surface \*psurface,PX\_FontModule \*mod,int x,int y,PX\_ALIGN align,const px\_char \*Text,px\_color Color); |
| 说明 | **绘制文本,注意,输入的字模必须是初始化指定编码的** |
| 参数 | Psurface 目标表面  X 原点x坐标  Y 原点y坐标  Text 文本  Color字颜色  Mod 字模库,如果这个参数为PX\_NULL,表示以默认字模库进行绘制(仅支持ANSI字符)  Align 字体对齐模式  PX\_ ALIGN\_LEFTTOP, 左上角对齐  PX\_ ALIGN\_MIDTOP,居中顶部对齐  PX\_ ALIGN\_RIGHTTOP,右上角对齐  PX\_ ALIGN\_LEFTMID,靠左居中对齐  PX\_ ALIGN\_CENTER,中心对齐  PX\_ ALIGN\_RIGHTMID,靠右居中对齐  PX\_ ALIGN\_LEFTBOTTOM,靠左底部对齐 |
| 返回值 | 绘制文本的像素宽度 |

#### 字模初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_FontModuleInitialize(px\_memorypool \*mp,PX\_FontModule \*module) |
| 说明 | **初始化一个字模库** |
| 参数 | Mp 内存池  Module 字模库 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 加载字模

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_FontModuleLoad(PX\_FontModule \*module,px\_byte \*buffer,px\_int size); |
| 说明 | **加载pxf字模到字模库,重复的字模以之前加载的为准** |
| 参数 | module 字模库  buffer pxf数据  size pxf数据大小 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 释放字模库

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_FontModuleFree(PX\_FontModule \*module); |
| 说明 | **释放字模库** |
| 参数 | module 字模库 |
| 返回值 | - |

#### 绘制单个字

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_FontModuleDrawCharacter(px\_surface \*psurface,PX\_FontModule \*mod,int x,int y,const px\_dword code,px\_color Color); |
| 说明 | **绘制单个字** |
| 参数 | Psurface 渲染表面  mod字模库  x ,y 左上角坐标  code 字编码  color 颜色 |
| 返回值 | - |

#### 计算绘制文本的宽高

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_FontModuleTextGetRenderWidthHeight(PX\_FontModule \*module,const px\_char \*Text,px\_int \*advance,px\_int \*height); |
| 说明 | **计算绘制文本的宽高** |
| 参数 | module字模库  Text 文本  Advance 步进长度  Height 绘制高度 |
| 返回值 | - |

### 信号处理

#### 离散傅里叶变换（DFT）

**复数结构体**

typedef struct \_\_complex

{

float re;// really

float im;// imaginary

}px\_complex;

|  |  |
| --- | --- |
| 函数名 | void PX\_DFT(\_IN complex x[],\_OUT complex X[],int N); |
| 说明 | **对复数信号进行DFT正变换** |
| 参数 | x 复信号  X 变换结果  N 复信号长度 |

|  |  |
| --- | --- |
| 函数名 | void PX\_IDFT(\_IN complex X[],\_OUT complex x[],int N); |
| 说明 | **对复数信号进行IDFT即逆变换** |
| 参数 | X 复信号  x 逆变换结果  N 复信号长度 |

#### 离散余弦变换（DCT）

**复数结构体**

|  |  |
| --- | --- |
| 函数名 | void PX\_DCT(\_IN px\_double x[],\_OUT px\_double X[],int N); |
| 说明 | **对信号进行DCT正变换** |
| 参数 | x 复信号  X 变换结果  N 复信号长度 |

|  |  |
| --- | --- |
| 函数名 | void PX\_IDFT(\_IN px\_double X[],\_OUT px\_double x[],int N); |
| 说明 | **对信号进行IDCT即逆变换** |
| 参数 | X 复信号  x 逆变换结果  N 复信号长度 |

#### 快速傅里叶变换（FFT）

|  |  |
| --- | --- |
| 函数名 | void PX\_FFT(\_IN complex x[],\_OUT complex X[],int N); |
| 说明 | **对复数信号进行FFT正变换** |
| 参数 | x 复信号  X 变换结果  N 复信号长度（必须是2基数） |

|  |  |
| --- | --- |
| 函数名 | void PX\_IFFT(\_IN complex X[],\_OUT complex x[],int N); |
| 说明 | **对复数信号进行IFFT即逆变换** |
| 参数 | X 复信号  x 逆变换结果  N 复信号长度（必须是2基数） |

|  |  |
| --- | --- |
| 函数名 | void PX\_FFT\_2(\_IN complex x[],\_OUT complex X[],int N\_N); |
| 说明 | **对复数信号进行二维FFT正变换** |
| 参数 | x二维复信号矩阵  X 变换结果  N\_N 复信号边长（复信号必须是2基边长的正方矩阵） |

|  |  |
| --- | --- |
| 函数名 | void PX\_IFFT\_2(\_IN complex X[],\_OUT complex x[],int N\_N); |
| 说明 | **对复数信号进行IFFT即逆变换** |
| 参数 | X 二维复信号矩阵  x 逆变换结果  N\_N 复信号边长（复信号必须是2基边长的正方矩阵） |

|  |  |
| --- | --- |
| 函数名 | void PX\_FFT\_2\_Shift(\_IN complex \_in[],\_OUT complex \_out[],int N\_N); |
| 说明 | **对复数信号进行FFTShift** |
| 参数 | \_in 二维复信号矩阵  \_out 逆变换结果  N\_N 复信号边长（复信号必须是2基边长的正方矩阵） |

#### 强制共轭对称

|  |  |
| --- | --- |
| 函数名 | void PX\_FT\_Symmetry(\_IN px\_complex X[],\_OUT px\_complex x[],px\_int N); |
| 说明 | **对一个傅里叶变换频域信号强制共轭对称(参考前半段数据)** |
| 参数 | x输入信号  X 输出信号  N 长度 |

#### 上采样(插值, UpSampled)

|  |  |
| --- | --- |
| 函数名 | void PX\_UpSampled(\_IN px\_complex x[],\_OUT px\_complex X[],px\_int N,px\_int L) |
| 说明 | **对一个信号进行上采样** |
| 参数 | x输入信号  X 输出信号  N 原长度  L 上采样倍数(1/M)  注意:输出数据缓存的长度应该至少是原长度的2\*L倍  如果原信号或输出信号长度不是2基数,将使用较慢的DFT进行上采样 |

#### 下采样(抽取, SubSampled, DownSampled)

|  |  |
| --- | --- |
| 函数名 | void PX\_DownSampled (\_IN px\_complex x[],\_OUT px\_complex X[],px\_int N,px\_int M) |
| 说明 | **对一个信号进行下采样(SubSampled)** |
| 参数 | x输入信号  X 输出信号  N 原长度  M 下采样倍数(1/M)  注意:输出数据缓存的长度应该至少是原长度的两倍  如果原信号或输出信号长度不是2基数,将使用较慢的DFT进行下采样 |

#### tukey窗

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WindowFunction\_tukey(px\_double data[],px\_int N); |
| 说明 | **构造一个tukey窗函数** |
| 参数 | data 数据缓存  N 长度 |
| 返回值 |  |

#### triangular窗

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WindowFunction\_ triangular (px\_double data[],px\_int N); |
| 说明 | **构造一个triangular窗函数** |
| 参数 | data 数据缓存  N 长度 |
| 返回值 |  |

#### blackMan窗

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WindowFunction\_blackMan(px\_double data[],px\_int N); |
| 说明 | **构造一个blackman窗函数** |
| 参数 | data 数据缓存  N 长度 |
| 返回值 |  |

#### hamming窗

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WindowFunction\_ hamming (px\_double data[],px\_int N); |
| 说明 | **构造一个hamming窗函数** |
| 参数 | data 数据缓存  N 长度 |
| 返回值 |  |

#### hanning窗

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WindowFunction\_ hanning (px\_double data[],px\_int N); |
| 说明 | **构造一个hanning窗函数** |
| 参数 | data 数据缓存  N 长度 |
| 返回值 |  |

#### kaiser窗

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WindowFunction\_kaiser(px\_double beta,px\_double data[],px\_int N); |
| 说明 | **构造一个kaiser窗函数** |
| 参数 | beta 就是字面的意思  data 数据缓存  N 长度 |
| 返回值 |  |

#### 应用窗函数

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WindowFunction\_Apply(px\_double data[],px\_double window[],px\_int N); |
| 说明 | **将一个窗函数应用于离散信号** |
| 参数 | data 数据缓存  window 窗函数缓存  N 长度 |
| 返回值 |  |

#### 计算系统幅频响应和相频响应

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_gain(px\_double b[],px\_double a[],px\_int m,px\_int n,px\_double x[],px\_double y[],px\_int len,px\_int sign); |
| 说明 | **计算系统幅频响应和相频响应** |
| 参数 | b存放滤波器分子多项式系数b[i],长度为m+1  a存放滤波器分母多项式系数a[i],长度为n+1  m 滤波器分子多项式阶数  n滤波器分母多项式阶数  x,y 输出数组,长度为len,当sign为1时存放滤波器幅频响应|H(w)|(x为实部,y为虚部),当sign为2时存放分贝表示的滤波器幅频相频  len 见上一项说明  sign 见上一项说明 |
| 返回值 |  |

#### 计算级联型系统幅频响应和相频响应

|  |  |
| --- | --- |
| 函数名 | px\_double b[],px\_double a[],px\_int n,px\_int ns,px\_double x[],px\_double y[],px\_int len,px\_int sign |
| 说明 | **计算级联型系统幅频响应和相频响应** |
| 参数 | b存放滤波器分子多项式系数b[i],矩阵,体积为ns\*(n+1)  a存放滤波器分母多项式系数a[i],矩阵,体积为ns\*(n+1)  n级联滤波器每节多项式阶数  ns 级联滤波器n阶节数L  x,y 输出数组,长度为len,当sign为1时存放滤波器幅频响应|H(w)|(x为实部,y为虚部),当sign为2时存放分贝表示的滤波器幅频相频  len 见上一项说明  sign 见上一项说明 |
| 返回值 |  |

#### 倒谱

|  |  |
| --- | --- |
| 函数名 | void PX\_Cepstrum(\_IN px\_complex x[],\_OUT px\_complex X[],px\_int N, PX\_CEOSTRUM\_TYPE type); |
| 说明 | **计算一个信号的倒谱域** |
| 参数 | x输入信号  X倒谱域  N信号长度  Type 倒谱类型, PX\_CEOSTRUM\_TYPE\_COMPLEX表示复倒谱  PX\_CEOSTRUM\_TYPE\_REAL 表示实倒谱 |
| 返回值 |  |

#### 人声基音频率估算

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_PitchEstimation(\_IN px\_complex x[],px\_int N,px\_int sampleRate); |
| 说明 | **估算人声基音频率** |
| 参数 | x输入信号  N信号长度  sampleRate 信号采样率 |
| 返回值 | 估算的基音频率 |

#### 估算瞬时频率

|  |  |
| --- | --- |
| 函数名 | px\_sine PX\_PhaseVocoder(px\_sine src,px\_double p2,px\_double delta\_t) |
| 说明 | **估算瞬时频率** |
| 参数 | src 原正弦波信号  p2 第二相位  delta\_t 取样时间 |
| 返回值 | 带有瞬时频率的正弦曲线 |

#### 预加重

|  |  |
| --- | --- |
| 函数名 | void PX\_PreEmphasise(const px\_double \*data, int len, px\_double \*out, px\_double preF) |
| 说明 | **对一个实信号进行预加重** |
| 参数 | data 输入信号  len 长度  out 输出预加重信号  preF 加重权值 范围为0.9-1.0 一般取0.97 |
| 返回值 |  |

#### 卡尔曼滤波

##### 初始化卡尔曼滤波器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_KalmanFilterInitialize(PX\_KalmanFilter \*filter,px\_double A,px\_double B,px\_double Q,px\_double H,px\_double R); |
| 说明 | **初始化卡尔曼滤波器** |
| 参数 | filter 卡尔曼滤波实例  A,B,Q,H,R参照卡尔曼滤波公式解释 |
| 返回值 |  |

##### 更新卡尔曼滤波器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_KalmanFilterUpdate(PX\_KalmanFilter \*filter,px\_double Zk,px\_double uk,px\_double wk); |
| 说明 | **初始化卡尔曼滤波器** |
| 参数 | filter 卡尔曼滤波实例  Zk 下一次更新值  uk,wk 参照公式 |
| 返回值 |  |

##### 预测值

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_KalmanFilterGetGuess(PX\_KalmanFilter \*filter) |
| 说明 | **取得卡尔曼滤波器预测值** |
| 参数 | filter 卡尔曼滤波实例 |
| 返回值 |  |

#### MFCC

##### 初始化MFCC滤波器组

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MFCCInitialize(PX\_MFCC \*mfcc,px\_int framesize,px\_int sampleRate,px\_int low,px\_int high); |
| 说明 | **初始化一个MFCC滤波器组** |
| 参数 | mfcc mfcc滤波器组  framesize 帧长度  samplerate 采样率  low hight 带通频率 |
| 返回值 |  |

##### MFCC滤波

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MFCCParse(PX\_MFCC \*mfcc,px\_double \*data,PX\_MFCC\_FEATURE \*out); |
| 说明 | **将数据通过MFCC滤波器组** |
| 参数 | mfcc mfcc滤波器组  data 帧数据，必须和帧长度匹配  out 输出的mfcc系数 |
| 返回值 | 如果满足一帧数据的parse返回PX\_TRUE，否者返回PX\_FALSE |

#### WAV文件格式

##### 取得wav声道数目

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_WAVEGetChannel(px\_byte \*buffer,px\_int size); |
| 说明 | **取得一个wav数据的声道数量** |
| 参数 | buffer指向wav数据指针  size wav数据长度 |
| 返回值 | 声道数量 |

##### 验证wav数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_WAVEVerify(px\_byte \*buffer,px\_int size); |
| 说明 | **验证一段数据是否是可以加载的wav数据** |
| 参数 | buffer指向wav数据指针  size wav数据长度 |
| 返回值 | 如果可以返回PX\_TRUE,否者返回PX\_FALSE |

##### 取得wav数据的PCM流大小

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_WAVEGetPCMSize(px\_byte \*buffer,px\_int size); |
| 说明 | **取得wav数据的PCM流大小** |
| 参数 | buffer指向wav数据指针  size wav数据长度 |
| 返回值 | 返回数据长度 |

#### 混音器

##### 初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SoundPlayInitialize(px\_memorypool \*mp, PX\_SoundPlay \*pSoundPlay); |
| 说明 | **初始化一个声音混音器** |
| 参数 | mp 内存池  pSoundPlay sound结构体 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

##### 设置用户自定义读函数

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SoundPlaySetUserRead(PX\_SoundPlay \*pSoundPlay,px\_void (\*userread)(px\_void \*userptr,px\_byte \*pBuffer,px\_int readSize),px\_void \*ptr); |
| 说明 | **用户自定义读函数** |
| 参数 | pSoundPlay sound结构体  userread 自定义读函数 |
| 返回值 |  |

##### 暂停/继续数据播放

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SoundPlayPause(PX\_SoundPlay \*pSoundPlay,px\_bool pause); |
| 说明 | **暂停/继续数据播放** |
| 参数 | pSoundPlay sound结构体  pause 为PX\_TRUE表示暂停,PX\_FALSE表示继续 |
| 返回值 |  |

##### 加载播放数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SoundAdd(PX\_SoundPlay \*pSound,PX\_Sound sounddata); |
| 说明 | **加载一个播放数据混音器中** |
| 参数 | pSoundPlay sound结构体  sounddata 播放数据 |
| 返回值 | 返回PX\_TRUE表示成功,否者返回PX\_FALSE |

##### 读取混音PCM数据流

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SoundRead(PX\_SoundPlay \*pSound,px\_byte \*pBuffer,px\_int readSize); |
| 说明 | **读取混音后的PCM流** |
| 参数 | pSoundPlay sound结构体  buffer PCM数据流  readsize 读取长度 |
| 返回值 | 返回PX\_TRUE表示成功,否者返回PX\_FALSE |

##### 释放混音器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SoundFree(PX\_SoundPlay \*pSound); |
| 说明 | **释放混音器** |
| 参数 | pSoundPlay sound结构体 |
| 返回值 | - |

##### 清除混音器所有音效

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SoundClear(PX\_SoundPlay \*pSound); |
| 说明 | **清除混音器所有音效** |
| 参数 | pSoundPlay sound结构体 |
| 返回值 | - |

##### 计算混音器音源个数

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_SoundPlayGetDataCount(PX\_SoundPlay \*pSoundPlay); |
| 说明 | **计算混音器音源个数** |
| 参数 | pSoundPlay sound结构体 |
| 返回值 | 混音器内音源个数 |

##### 创建音源实例

|  |  |
| --- | --- |
| 函数名 | PX\_Sound PX\_SoundCreate(PX\_SoundData \*data,px\_bool loop); |
| 说明 | **创建音源实例** |
| 参数 | PX\_SoundData 音源数据  loop 这个音源是否循环播放 |
| 返回值 | 音源实例 |

##### 加载静态音源

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SoundStaticDataCreate(PX\_SoundStaticData \*sounddata,px\_memorypool \*mp,px\_byte \*data,px\_int datasize); |
| 说明 | **加载一个静态音源,在使用结束后,这个音源需要手动释放** |
| 参数 | sounddata 静态音源  mp 存储内存池  data 原始解析数据(例如WAV格式)  data 原始解析数据大小 |
| 返回值 | 成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 释放静态音源

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SoundStaticDataFree(PX\_SoundStaticData \*sounddata); |
| 说明 | **释放静态音源** |
| 参数 | sounddata 静态音源  mp 存储内存池  data 原始解析数据(例如WAV格式)  data 原始解析数据大小 |
| 返回值 | 成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 拷贝静态音源数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SoundStaticDataCopy(px\_memorypool \*mp,PX\_SoundData \*resSounddata,PX\_SoundData \*targetSounddata); |
| 说明 | **拷贝一个静态音源,在使用结束后,这个音源需要手动释放** |
| 参数 | mp 存储内存池  resSounddata原始音源数据  targetSounddata目标音源数据 |
| 返回值 | 成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 调音台

##### 时域延拓

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_TuningTimeScale(px\_double timescale,px\_double in[],px\_int count,px\_memory \*out); |
| 说明 | **基于相位声码器的变速不变调TSM(时域压拓)系统** |
| 参数 | **timescale 时域压拓系数**  **in 输入信号(建议归一化处理)**  count 信号数量  out 输出信号,该内存变量需要被初始化 |
| 返回值 | 成功返回PX\_TRUE,否者PX\_FALSE |

##### 调音台初始化

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TuningInitialize(px\_memorypool \*mp,PX\_Tuning \*tuning,px\_double pitchShift,px\_double window[],px\_double filter[],px\_double fix[],PX\_TUNING\_WINDOW\_SIZE windowsize); |
| 说明 | **基于重采样和相位声码器的变调不变速系统调音台初始化** |
| 参数 | **mp 计算内存池**  **tuning 调音台指针**  **pitchshift 调变系数,例如音调提升2倍则为2,降低一倍则为0.5**  window 窗函数(建议为能量相等的处理窗,50% overlap)  filter 滤波器,长度等于windowsize指示长度  fix 增量值,长度等于windowsize指示长度(归一化后)  windowsize 窗函数类型  PX\_TUNING\_WINDOW\_SIZE\_8, 长度为8的窗,主要用于测试  PX\_TUNING\_WINDOW\_SIZE\_128, 长度为128的窗  PX\_TUNING\_WINDOW\_SIZE\_256, 长度为256的窗  PX\_TUNING\_WINDOW\_SIZE\_512, 长度为512的窗,建议值  PX\_TUNING\_WINDOW\_SIZE\_1024, 长度为1024的窗,建议值  PX\_TUNING\_WINDOW\_SIZE\_2048, 长度为2048的窗,建议值  PX\_TUNING\_WINDOW\_SIZE\_4096, 长度为4096的窗  PX\_TUNING\_WINDOW\_SIZE\_8192, 长度为8192的窗 |
| 返回值 |  |

##### 调音台滤波器系统

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_TuningFilter(PX\_Tuning \*tuning,\_IN px\_double frame\_unit[],px\_int Size,\_OUT px\_double out\_unit[]); |
| 说明 | **基于重采样和相位声码器的变调不变速系统调音台滤波器系统**  **\*该滤波器为FIR类型滤波器**  **\*输入信号必须是连续的** |
| 参数 | **tuning 调音台指针**  frame\_unit[] 输入数据(建议归一化处理)  Size 输入数据长度  out\_unit 输出数据 |
| 返回值 | 输出长度 |

##### 调音台TSM

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TuningTimeScale(px\_double timescale,px\_double in[],px\_int count,px\_memory \*out); |
| 说明 | **调音台TSM算法** |
| 参数 | **timescale 压拓系数**  **in 输入信号(归一化建议)**  **count 输入信号长度**  **out 输出内存** |
| 返回值 |  |

##### 设置调音台的PitchShift

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TuningSetPitchShift(PX\_Tuning \*tuning,px\_double pitchShift); |
| 说明 | **设置调音台的PitchShift** |
| 参数 | **tuning 调音台指针**  PitchShift 移调系数 |
| 返回值 |  |

##### 设置调音台的滤波器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TuningSetFilter(PX\_Tuning \*tuning,px\_double filter[]); |
| 说明 | **设置调音台的滤波器** |
| 参数 | **tuning 调音台指针**  filter滤波器系数 |
| 返回值 |  |

##### 设置调音台的频域补偿系数

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TuningSetFix(PX\_Tuning \*tuning,px\_double fix[]); |
| 说明 | **设置调音台的频域补偿系数** |
| 参数 | **tuning 调音台指针**  fix频域补偿系数 |
| 返回值 |  |

##### 设置调音台的ZCR阈值

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_TuningSetZCR(px\_double low,px\_double high); |
| 说明 | **设置调音台的ZCR阈值,当输入信号的ZCR不在阈值范围内时,将不对信号进行变调处理** |
| 参数 | **tuning 调音台指针**  low 最低ZCR  high 最高ZCR |
| 返回值 |  |

### 反向传播神经网络

PainterEngine 提供一个基础的反向传播神经网络框架,集成了linear tanh sigmod reLU等激活函数,regularzation方式及权值初始化方式.

#### 初始化神经网络框架

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ANNInitialize(px\_memorypool \*mp,PX\_ANN \*ann,px\_double learningRate,PX\_ANN\_REGULARZATION regularzation,px\_double regularization\_rate); |
| 说明 | **初始化一个反向传播神经网络框架** |
| 参数 | **mp 运行内存池**  **ann 神经网络结构**  **learningRate 学习率**  regularization 正则化方式  参考:  typedef enum  {  PX\_ANN\_REGULARZATION\_NONE,  PX\_ANN\_REGULARZATION\_L1,  PX\_ANN\_REGULARZATION\_L2  }PX\_ANN\_REGULARZATION;  regularization\_rate 正则化速率 |
| 返回值 | 返回PX\_TRUE表示成功,否者返回PX\_FALSE |

#### 添加网络层

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ANNAddLayer(PX\_ANN \*pAnn,px\_int Neurals,px\_double bias,PX\_ANN\_ACTIVATION\_FUNCTION activation,PX\_ANN\_LAYER\_WEIGHT\_INITMODE mode,px\_double weight\_c); |
| 说明 | **添加一个网络层(其中第一层为输入层,最后一层为输出层)** |
| 参数 | **ann 神经网络结构**  Neurals 神经元数量  bias 偏置  activation 激活函数类型,参照  typedef enum  {  PX\_ANN\_ACTIVATION\_FUNCTION\_SIGMOID, //sigmoid  PX\_ANN\_ACTIVATION\_FUNCTION\_TANH,//tanh  PX\_ANN\_ACTIVATION\_FUNCTION\_LINEAR,//linear  PX\_ANN\_ACTIVATION\_FUNCTION\_RELU,//ReLU  }PX\_ANN\_ACTIVATION\_FUNCTION;  mode 权值初始方式,参照  typedef enum  {  PX\_ANN\_LAYER\_WEIGHT\_INITMODE\_CONST, //常量  PX\_ANN\_LAYER\_WEIGHT\_INITMODE\_RAND,//随机数  PX\_ANN\_LAYER\_WEIGHT\_INITMODE\_GAUSSRAND,//正态分布随机数  }PX\_ANN\_LAYER\_WEIGHT\_INITMODE;  weight\_c 当权值初始方式为常量时的常量值 |
| 返回值 | 返回PX\_TRUE表示成功,否者返回PX\_FALSE |

#### 训练网络

|  |  |
| --- | --- |
| 函数名 | px\_double PX\_ANNTrain(PX\_ANN \*pAnn,px\_double \*input,px\_double \*expect); |
| 说明 | **监督学习训练网络** |
| 参数 | **ann 神经网络结构**  input 输入数组(与输入层神经元对应)  expect 期望输出数组(与输出层神经元对应) |
| 返回值 | train loss值 |

#### 前向传播

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ANNForward(PX\_ANN \*pAnn,px\_double \*input); |
| 说明 | **神经网络前向传播** |
| 参数 | **ann 神经网络结构**  input 输入数组(与输入层神经元对应) |
| 返回值 |  |

#### 取得输出结果

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ANNGetOutput(PX\_ANN \*pAnn,px\_double \*result); |
| 说明 | **神经网络前向传播的输出结果** |
| 参数 | **ann 神经网络结构**  result 输出数组(与输入层神经元对应) |
| 返回值 |  |

#### 重置神经网络

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ANNReset(PX\_ANN \*pANN); |
| 说明 | **重置神经网络的所有权重** |
| 参数 | **ann 神经网络结构** |
| 返回值 |  |

#### 释放神经网络框架

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ANNFree(PX\_ANN \*pAnn); |
| 说明 | **释放框架** |
| 参数 | **ann 神经网络结构** |
| 返回值 |  |

#### 导出神经网络框架

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ANNExport(PX\_ANN \*pAnn,px\_void \*buffer,px\_int \*size); |
| 说明 | **将神经网络当前的训练框架导出到内存结构** |
| 参数 | **ann 神经网络结构.**  **buffer 导出buffer(当该值为PX\_NULL时表示只计算大小)**  **size 导出数据大小** |
| 返回值 | 成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 导入神经网络框架

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ANNImport(px\_memorypool \*mp,PX\_ANN \*pAnn,px\_void \*buffer,px\_int size); |
| 说明 | **从内存结构中导入神经网络框架** |
| 参数 | **mp 神经网络使用内存池**  **ann 未初始化的神经网络结构.**  **buffer 导入buffer**  **size 导入数据大小** |
| 返回值 | 成功返回PX\_TRUE,否者返回PX\_FALSE |

### 物理

#### 四叉树AABB碰撞检测

PainterEngine Core提供四叉树AABB碰撞检测算法,该算法用于加速矩形box区域的碰撞检测,使用自定义的树深调节对

##### 初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_QuadtreeCreate(px\_memorypool \*mp,PX\_Quadtree \*pQuadtree,px\_float mapStartX,px\_float mapStartY,px\_float mapWidth,px\_float mapHeight,px\_int ObjectsCount,px\_int deep); |
| 说明 | **创建四叉树根结构** |
| 参数 | mp 一次性内存池,该内存池不能够存在其他的节点,仅用于四叉树计算使用  pQuadtree, 四叉树结构体  mapStartX 需要分隔的域左上角起始坐标X  mapStartY 需要分隔的域左上角起始坐标Y  mapWidth,mapHeight 域的宽度和高度  ObjectsCount 需要计算的节点数  deep 深度,建议在2-4范围,启始深度为0,每个深度分4个区域 |
| 返回值 | 如果成功返回PX\_TRUE,否者为PX\_FALSE |

##### 添加节点

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_QuadtreeAddNode(PX\_Quadtree \*pQuadtree,px\_float CenterX,px\_float CenterY,px\_float width,px\_float height,PX\_Quadtree\_UserData userData); |
| 说明 | **在四叉树添加一个节点,该节点会不参与碰撞计算** |
| 参数 | pQuadtree 四叉树结构体  CenterX, CenterY,width,height,AABB描述  userData 碰撞测试的用户数据,这个数据将被附加到碰撞结果中,注意,相同的userData将被认为是重复数据 |

##### 碰撞测试

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_QuadtreeTestNode(PX\_Quadtree \*pQuadtree,px\_float x,px\_float y,px\_float width,px\_float height,PX\_Quadtree\_UserData userData) |
| 说明 | **在四叉树设置接参与碰撞计算的节点,这个节点不会被加入四叉树中** |
| 参数 | pQuadtree 四叉树结构体  x,y,z,width,height,length AABB box  userData该节点的标识号,该索引将出现在碰撞结果中,注意,如果节点的ptr相同,则该节点将标记为重复节点不计入碰撞检测中 |

##### 重置碰撞表

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_QuadtreeResetTest(PX\_Quadtree \*pQuadtree); |
| 说明 | **重置四叉树的碰撞表** |
| 参数 | pQuadtree 四叉树结构体 |

### 密码学

#### AES

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AES\_Initialize(PX\_AES \*aesStruct,PX\_AES\_KeySize keySize,AES\_BYTE keyByte[]); |
| 说明 | **初始化AES编码器** |
| 参数 | aseStruct AES编码器  keySize 128,192,256加密  keyByte 对应位数长度密钥 |
|  |  |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AES\_Cipher(PX\_AES \*aesStruct,void \*input, void \*output); |
| 说明 | **编码16字节的数据** |
| 参数 | aseStruct AES编码器  input 输入  output 输出 |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AES\_CipherBuffer(PX\_AES \*aesStruct,void \*input, px\_int size,void \*output); |
| 说明 | **编码一段的数据,这个数据的长度必须是16的整数倍** |
| 参数 | aseStruct AES编码器  input 输入  size 长度  output 输出 |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AES\_InvCipher(PX\_AES \*aesStruct,void \*input, void \*output); |
| 说明 | **解码16字节的数据** |
| 参数 | aseStruct AES编码器  input 输入  output 输出 |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AES\_InvCipherBuffer(PX\_AES \*aesStruct,void \*input, px\_int size,void \*output); |
| 说明 | **解码一段的数据,这个数据的长度必须是16的整数倍** |
| 参数 | aseStruct AES编码器  input 输入  size 长度  output 输出 |

#### curve25519

curve25519 是ECC加密衍生的Diffie-Hellman函数,用于生成一个公共密钥保证通讯安全

|  |  |
| --- | --- |
| 函数名 | px\_void curve25519\_donna(px\_byte \*mypublic, const px\_byte \*secret, const px\_byte \*basepoint); |
| 说明 | **解码16字节的数据** |
| 参数 | mypublic 32字节的公钥,输出  secret 32字节私钥,输入  basepoint 32字节长度的基点  私钥需要如下处理  mysecret[0] &= 248;  mysecret[31] &= 127;  mysecret[31] |= 64;  基点可以为  static const px\_byte basepoint[32] = {9}; |

#### SHA256

|  |  |
| --- | --- |
| 函数名 | PX\_Sha256Calculate (void\*Buffer,px\_uint32 BufferSize, PX\_SHA256\_HASH\* Digest ); |
| 说明 | **Sha256** |
| 参数 | buffer 输入缓存  buffersize 缓存大小  digest 输出hash值 |

### 编码及数据压缩

#### BASE64

##### 编码

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_Base64Encode(const px\_byte \*in, px\_uint inlen, px\_char \*out); |
| 说明 | **使用Base64编码一段数据** |
| 参数 | in 输入数据缓存  input\_size 输入数据长度  \_out 输出数据缓存,不可为NULL,可以使用PX\_Base64GetEncodeLen预先计算编码后长度 |
| 返回值 | 编码长度 |

##### 解码

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_Base64Decode(const px\_char \*\_in, px\_uint input\_size, px\_byte \*out); |
| 说明 | **使用Base64解码一段数据** |
| 参数 | in 输入数据缓存  input\_size 输入数据长度  \_out 输出数据缓存,不可为NULL,可以使用PX\_Base64GetDecodeLen预先计算解码后长度 |
| 返回值 | 解码长度,如果为0表示无法解码 |

##### 长度计算

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_Base64GetEncodeLen(px\_uint codeLen);  px\_uint PX\_Base64GetDecodeLen(px\_uint codeLen); |
| 说明 | **计算BASE64的编码/解码长度** |
| 参数 | Codelen 输入数据长度 |
| 返回值 | 编码/解码后的长度,如果为0表示无法编/解码 |

#### ARLE

ARLE(AdvanceRun Length Encoding)改进行程长度压缩算法,主要适用于图像及音频数据的压缩,最坏情况下,长度为N\*4/3,最优秀情况下,为长度的N\*2/127

##### 压缩

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ArleCompress(px\_byte \*\_in,px\_uint input\_size,px\_byte \*\_out,px\_uint \*\_outsize); |
| 说明 | **使用ARLE压缩一段数据** |
| 参数 | \_in 输入数据缓存  input\_size 输入数据长度  \_out 输出数据缓存,可以为NULL表示仅计算输出长度  \_outsize 输出长度 |

##### 解压缩

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ArleCompress(px\_byte \*\_in,px\_uint input\_size,px\_byte \*\_out,px\_uint \*\_outsize); |
| 说明 | **使用ARLE压缩一段数据** |
| 参数 | \_in 输入数据缓存  input\_size 输入数据长度  \_out 输出数据缓存,可以为NULL表示仅计算输出长度  \_outsize 输出长度 |

#### Huffman

##### 压缩

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_HuffmanCompress(px\_byte \*\_in,px\_uint input\_size,px\_byte \*\_out,px\_uint \*\_outsize); |
| 说明 | **使用Huffman压缩一段数据** |
| 参数 | \_in 输入数据缓存  input\_size 输入数据长度  \_out 输出数据缓存,可以为NULL表示仅计算输出长度  \_outsize 输出长度 |

##### 解压缩

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Huffmandecompress(px\_byte \*\_in,px\_uint input\_size,px\_byte \*\_out,px\_uint \*\_outsize); |
| 说明 | **使用Huffman压缩一段数据** |
| 参数 | \_in 输入数据缓存  input\_size 输入数据长度  \_out 输出数据缓存,可以为NULL表示仅计算输出长度  \_outsize 输出长度 |

### 3D渲染管线

PainterEngine提供一个轻量级软件3D渲染管线,其中大部分参数都以默认方式部署(质量优先),可用于3D动画渲染,静态模型渲染,但因为渲染性能消耗较大,并不建议使用该3D渲染管线直接开发需要动态渲染的复杂游戏,3D渲染管线是作为一个模块存在于PainterEngine中的,PainterEngine并不仅作为3D渲染器,不建议将3D渲染模式作为PainterEngine中的主要使用模式中.

##### 世界矩阵

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_WorldInitialize(PX\_3D\_World \*world,px\_float x,px\_float y,px\_float z,px\_float rotX,px\_float rotY,px\_float rotZ,px\_float scale); |
| 说明 | **初始化一个世界坐标,在渲染中该世界矩阵将直接作用于渲染列表** |
| 参数 | world 世界矩阵  x,y,z 世界矩阵的平移坐标  rotX,rotY,rotZ 绕X,Y,Z轴的旋转角度  scale 缩放比例 |

##### 欧拉相机矩阵

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_CameraEulerInitialize(px\_memorypool \*mp,PX\_3D\_Camera \*camera,px\_point4D cameraPosition,px\_float rotX,px\_float rotY,px\_float rotZ,px\_float near,px\_float far,px\_float fov,px\_float viewPortWidth,px\_float viewPortHeight); |
| 说明 | **初始化一个欧拉相机矩阵,在渲染中该矩阵将直接作用于渲染列表** |
| 参数 | mp 内存池,用于分配Z缓存  camera 相机矩阵  camera 相机位置  rotX,rotY,rotZ 绕X,Y,Z轴的旋转角度  far 远裁剪面  near 近裁剪面,这个值不能小于1.0  fov 视锥角度  viewportWidth 窗口宽度(像素为单位)  viewPortHeight 窗口高度(像素为单位) |

##### UVN相机矩阵

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_CameraUVNInitialize(px\_memorypool \*mp,PX\_3D\_Camera \*camera,px\_point4D cameraPosition,px\_point4D cameraTarget,px\_float near,px\_float far,px\_float fov,px\_float viewPortWidth,px\_float viewPortHeight); |
| 说明 | **初始化一个UVN相机矩阵,在渲染中该矩阵将直接作用于渲染列表** |
| 参数 | mp 内存池,用于分配Z缓存  camera 相机矩阵  cameraposition 相机位置  target 相机指向位置  far 远裁剪面  near 近裁剪面,这个值不能小于1.0  fov 视锥角度  viewportWidth 窗口宽度(像素为单位)  viewPortHeight 窗口高度(像素为单位) |

##### 设置UVN相机位置

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_CameraSetPosition(PX\_3D\_Camera \*camera,px\_point4D cameraPosition,px\_point4D cameraTarget); |
| 说明 | **设置uvn相机位置** |
| 参数 | cameraposition 相机位置  target 相机指向位置 |

##### 释放相机矩阵

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_CameraFree(PX\_3D\_Camera \*camera) |
| 说明 | **释放相机矩阵** |
| 参数 | camera 相机矩阵 |

##### 渲染列表

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_3D\_RenderListInitialize(px\_memorypool \*mp,PX\_3D\_RenderList \*list,px\_dword PX\_3D\_PRESENTMODE,PX\_3D\_CULLMODE cullmode,px\_texture \*ptexture); |
| 说明 | **渲染列表是PainterEngine渲染管线的基本渲染单元**  **所有的面都必须加载到该渲染列表中进行渲染** |
| 参数 | mp 使用内存池  list 渲染列表  PX\_3D\_PRESENTMODE 显示方式  包括:  #define PX\_3D\_PRESENTMODE\_LINE 1 线段渲染  #define PX\_3D\_PRESENTMODE\_TEXTURE 2 纹理渲染  #define PX\_3D\_PRESENTMODE\_PURE 4 纯色渲染  cullmode 裁剪模式  PX\_3D\_CULLMODE\_NONE, 不裁剪  PX\_3D\_CULLMODE\_CW, 顺时针裁剪  PX\_3D\_CULLMODE\_CCW, 逆时针裁剪  ptexture 纹理 |

##### 渲染列表添加三角形面

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_3D\_RenderListPush(PX\_3D\_RenderList \*list,PX\_3D\_Face face); |
| 说明 | **将一个三角形面添加到渲染列表当中** |
| 参数 | list 渲染列表  face 三角形面  typedef struct  {  px\_point4D position;//顶点位置  px\_vector4D normal;//法线(由渲染管线计算)  px\_color clr;//颜色  px\_float u,v;//uv纹理映射坐标  }PX\_3D\_Vertex;  typedef struct  {  px\_dword state;//由渲染管线控制  PX\_3D\_Vertex vertex[3];//顶点  PX\_3D\_Vertex transform\_vertex[3];//变换后顶点,由渲染管线控制  }PX\_3D\_Face;  每次使用PX\_3D\_Face建议对其内存清0 |

##### 推算凹凸纹理

|  |  |
| --- | --- |
| 函数名 | px\_point \*PX\_3D\_CreateTextureNormal(px\_memorypool \*mp,px\_texture \*pTexture); |
| 说明 | **以明暗关系推算一张纹理的法向量** |
| 参数 | Mp 法向量矩阵分配空间内存池  pTexture 需要计算的纹理 |
| 返回值 | 和纹理等高\*宽的法向量数组 |

##### 推算高度图的凹凸纹理

|  |  |
| --- | --- |
| 函数名 | px\_point \*PX\_3D\_CreateBumpTextureNormal(px\_memorypool \*mp,px\_texture \*pTexture); |
| 说明 | **以高度图推算其凹凸纹理** |
| 参数 | Mp 法向量矩阵分配空间内存池  pTexture 需要计算的高度图纹理 |
| 返回值 | 和纹理等高\*宽的法向量数组 |

##### 渲染列表矩阵变换

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_RenderListTransform(PX\_3D\_RenderList \*list,px\_matrix mat,PX\_3D\_RENDERLIST\_TRANSFORM t); |
| 说明 | **将渲染列表中每一个顶点都使用mat矩阵变换** |
| 参数 | list 渲染列表  mat 变换矩阵  PX\_3D\_RENDERLIST\_TRANSFORM  PX\_3D\_RENDERLIST\_TRANSFORM\_LOCAL\_TO\_LOCAL,变换原始顶点并替换原始顶点  PX\_3D\_RENDERLIST\_TRANSFORM\_LOACL\_TO\_GLOBAL,变换原始顶点,并将结果放置在变换顶点中  PX\_3D\_RENDERLIST\_TRANSFORM\_GLOBAL\_TO\_GLOBAL,变换变换顶点,并将结果放置在变换顶点中 |

##### 渲染列表清空

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_RenderListReset(PX\_3D\_RenderList \*list); |
| 说明 | **清空渲染列表,注意:该函数并不清理内存,该列表仍然可用** |
| 参数 | list 渲染列表 |

##### 释放列表清空

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_RenderListFree(PX\_3D\_RenderList \*list); |
| 说明 | **释放渲染列表,并清理内存,该列表将不可用** |
| 参数 | list 渲染列表 |

##### 开始渲染

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_Scene(PX\_3D\_RenderList \*list,PX\_3D\_World \*world,PX\_3D\_Camera \*camera) |
| 说明 | **开始渲染(在Present之前调用)** |
| 参数 | list 渲染列表  word 世界矩阵  camera 相机矩阵 |

##### 渲染列表

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_Present(px\_surface \*psurface, PX\_3D\_RenderList \*list); |
| 说明 | **渲染列表到surface** |
| 参数 | psurface 渲染表面  list 渲染列表  world 世界矩阵  camera 相机矩阵 |

##### 设置PixelShader

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_RenderListSetPixelShader(PX\_3D\_RenderList \*list,PX\_3D\_PixelShader func); |
| 说明 | **设置渲染流水线的PixelShader** |
| 参数 | list 渲染列表  func PixelShader回调函数 |

## Kernel

### 词法分析机

PainterEngine Kernel 提供通用词法分析机,用于对常用文本类型的脚本,数据格式语言的词法分析,在使用词法分析机之前,都应该使用PX\_LexerInit对词法分析机进行初始化操作,在词法分析机使用完毕后,应该使用PX\_LexerFree释放其占用资源.

#### 初始化词法分析机

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LexerInit(px\_lexer \*lexer,px\_memorypool \*mp); |
| 说明 | **初始化词法分析机** |
| 参数 | lexer px\_lexer结构指针  mp 词法分析内存池 |

#### 注册注释块

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LexerRegisterComment(px\_lexer \*lexer,px\_char Begin[],px\_char End[]); |
| 说明 | **注册注释块** |
| 参数 | lexer px\_lexer结构指针  begin 注释块的起始字符串  end 注释块的结束字符串 |
| 备注 | 如果注释块的结束字符串是回车,则该注释块在解析后为一个回车符 |

#### 注册包含块

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_LexerRegisterContainer(px\_lexer \*lexer,px\_char Begin[],px\_char End[]); |
| 说明 | **注册注释块** |
| 参数 | lexer px\_lexer结构指针  begin 包含块的起始字符串  end 包含块的结束字符串 |
| 返回值 | 该包含块的索引ID |

#### 注册空白符

多个连续的空白符在解析过程中将被解析为单个空白符

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LexerRegisterSpacer(px\_lexer \*lexer,px\_char Spacer); |
| 说明 | **注册空白符** |
| 参数 | lexer px\_lexer结构指针  Spacer 空白符 |

#### 注册分隔符

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_LexerRegisterDelimiter(px\_lexer \*lexer,px\_char Delimiter); |
| 说明 | **注册分隔符** |
| 参数 | lexer px\_lexer结构指针  Delimiter分隔符 |
| 返回值 | 该分隔符的ID |

#### 取得分隔符ID

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_LexerGetDelimiterType(px\_lexer \*lexer,px\_char Delimiter); |
| 说明 | **取得分隔符ID** |
| 参数 | lexer px\_lexer结构指针  Delimiter分隔符 |
| 返回值 | 该分隔符的注册ID |

#### 取得字符串包含块类型

|  |  |
| --- | --- |
| 函数名 | px\_uint PX\_LexerGetContainerType(px\_lexer \*lexer,px\_char \*pContainerText); |
| 说明 | **取得字符串包含块类型** |
| 参数 | lexer px\_lexer结构指针  pContainerText解释文本 |
| 返回值 | 该包含块的注册ID |

#### 释放词法分析机资源

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LexerFree(px\_lexer \*lexer); |
| 说明 | **释放词法分析机资源** |
| 参数 | lexer px\_lexer结构指针 |

#### 加载分析文本

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LexerSortText(px\_lexer \*lexer,px\_char \*SourceText);  px\_bool PX\_LexerLoadSourceFromMemory(px\_lexer \*lexer,px\_byte \*buffer); |
| 说明 | **释放词法分析机资源** |
| 参数 | lexer px\_lexer结构指针  SourceText Buffer 分析文本 |
| 返回值 | 如果成功解析返回PX\_TRUE,否则返回PX\_FALSE(词法错误) |

#### 读取文本

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LexerReadString(px\_lexer \*lexer,px\_string \*str,px\_uint size); |
| 说明 | **在当前词法分析状态下读取文本,该文本不置入词法分析** |
| 参数 | lexer px\_lexer结构指针  str 字符串指针  size 大小 |
| 返回值 | 如果成功解析返回PX\_TRUE,否则返回PX\_FALSE(词法错误) |

#### 取得特殊符号

|  |  |
| --- | --- |
| 函数名 | px\_char PX\_LexerGetSymbol(px\_lexer \*lexer); |
| 说明 | **取得当前分析的符号(分隔符,空白符或回车符)** |
| 参数 | lexer px\_lexer结构指针 |
| 返回值 | 当前分析的符号 |

#### 取得当前词

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LexerGetLexemeString(px\_lexer \*lexer,px\_string \*str); |
| 说明 | **取得当前词文本** |
| 参数 | lexer px\_lexer结构指针  str px\_string指针 |
| 返回值 |  |

可以通过访问px\_lexer-> CurLexeme 直接取得词字符串

#### 取得包含文本

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LexerGetIncludedString(px\_lexer \*lexer,px\_string \*str); |
| 说明 | **如果当前词为包含类型,则取得其包含文本** |
| 参数 | lexer px\_lexer结构指针  str px\_string指针 |
| 返回值 |  |

#### 设置词大小写敏感

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LexerSetTokenCase(px\_lexer \*lexer,PX\_LEXER\_LEXEME\_CASE \_case); |
| 说明 | **设置返回词大小写,为PX\_LEXER\_LEXEME\_CASE枚举类型,如果被设置,读取的词除包含类型外都会转换为对应大小写** |
| 参数 | lexer px\_lexer结构指针  \_case大小写类型 |
| 返回值 |  |

#### 取得词法分析机当前状态

|  |  |
| --- | --- |
| 函数名 | PX\_LEXER\_STATE PX\_LexerGetState(px\_lexer \*lexer); |
| 说明 | **取得当前词法分析机状态,可以在之后重设该状态** |
| 参数 | lexer px\_lexer结构指针 |
| 返回值 | 词法分析机当前状态 |

#### 设置词法分析机当前状态

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LexerRestoreState(PX\_LEXER\_STATE state); |
| 说明 | **重新设置词法分析机状态** |
| 参数 | state 词法分析机状态结构体 |
| 返回值 | - |

#### 判断当前词是否是合法数字

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LexerIsLememeIsNumeric(px\_lexer \*lexer); |
| 说明 | **判断当前词是否是合法数字** |
| 参数 | lexer px\_lexer结构指针 |
| 返回值 | 如果是返回PX\_TRUE如果不是返回PX\_FALSE |

#### 读取下一个字母

|  |  |
| --- | --- |
| 函数名 | px\_char PX\_LexerGetNextChar(px\_lexer \*lexer); |
| 说明 | **直接读取下一个字母** |
| 参数 | lexer px\_lexer结构指针 |
| 返回值 | 下一个字母 |

#### 读取下一个词

|  |  |
| --- | --- |
| 函数名 | PX\_LEXER\_LEXEME\_TYPE PX\_LexerGetNextLexeme(px\_lexer \*lexer); |
| 说明 | **直接读取下一个词** |
| 参数 | lexer px\_lexer结构指针 |
| 返回值 | 该词的词性,参考PX\_LEXER\_LEXEME\_TYPE  PX\_LEXER\_LEXEME\_TYPE\_END =0, 结束  PX\_LEXER\_LEXEME\_TYPE\_SPACER =1,空白符  PX\_LEXER\_LEXEME\_TYPE\_DELIMITER =2,分割符  PX\_LEXER\_LEXEME\_TYPE\_CONATINER =3,包含块  PX\_LEXER\_LEXEME\_TYPE\_NEWLINE =4,换行符  PX\_LEXER\_LEXEME\_TYPE\_TOKEN =5,词块  PX\_LEXER\_LEXEME\_TYPE\_ERR = -1,错误的词性 |

#### 取得当前词性

|  |  |
| --- | --- |
| 函数名 | PX\_LEXER\_LEXEME\_TYPE PX\_LexerGetCurrentLexeme(px\_lexer \*lexer); |
| 说明 | **取得当前词词性** |
| 参数 | lexer px\_lexer结构指针 |
| 返回值 | 该词的词性,参考PX\_LEXER\_LEXEME\_TYPE  PX\_LEXER\_LEXEME\_TYPE\_END =0, 结束  PX\_LEXER\_LEXEME\_TYPE\_SPACER =1,空白符  PX\_LEXER\_LEXEME\_TYPE\_DELIMITER =2,分割符  PX\_LEXER\_LEXEME\_TYPE\_CONATINER =3,包含块  PX\_LEXER\_LEXEME\_TYPE\_NEWLINE =4,换行符  PX\_LEXER\_LEXEME\_TYPE\_TOKEN =5,词块  PX\_LEXER\_LEXEME\_TYPE\_ERR = -1,错误的词性 |

### PainterScript语法规则

#### 脚本语法

Painter Script是PainterEngine所支持下的编译型脚本,是一个文件系统无关的支持脚本.

Painter Script以下简称为SS,其具有如下的语法特性.

Painter Script是大小写无关的语言.

Painter Script是文件系统无关的语言,所有SS脚本都不以文件系统作为标识,所有的SS源文件必须以一个#name “id”作为标识

其中id可以为任意合法字符,在一个Painter Script项目中,这个id必须是唯一的

Painter Script是一个弱类型语言,类型有 整数 浮点数 字符串类型 内存类型

Painter Script使用双引号表示一个字符串类型

Painter Script使用单引号表示一个字符类型

SS 使用大括号表示一个内存类型

SS 以分号作为语句结束

以下以BNF文法来描述Painter Script语法结构

##### 语法规则(BNF文法)

定义

Token::=符合命名规范的词或数字;

vName::= 符合命名规范的词

Numeric::=一个合法的数字,包括以0x开头的16进制数(最大不超过32位)

##### 预处理

###### #define name Token

#define用于将一个标识符替换为另一个标识符

所有name 类型都会被替换为Token

<预定义>::=”#define” <Token> <Token>

例如

#define PI 3.14159265

表示之后的PI都会被替换为3.14159265

###### #include “name”

#include用于将一个以name命名的源文件替换到当前预定义处

#include\_STATEMENT::= “#include” <”name”>

###### #runtime thread count

#runtime thread 用于定义最大支持线程数量

###### #runtime stack count

#runtime stack 用于定义默认的栈大小

##### 注释

PainterScript的注释方式与C语言一致

一种为双斜杠,代表本行注释,例如,下面是一个合法的注释

//note

同时,PainterScript支持/\*\*/类型注释.例如下文也是一个合法的注释

/\*note

note

\*/

##### 常量

PainterScript有以下几种常量

###### 数字

例如123,456及3.14159都是合法的数字常量,同时,PainterScript支持十六进制数常量

例如0x01,0xffffffff,其大小不应该超过32位数的定义

最后,字符ascii的表达方式也被解释为数字,例如

‘a’表示a的ascii码,其也作为一种数字常量

###### 字符串

用引号来包含一个字符串运算,例如

“hello world”

“Painter Script”

都是一个合法的字符串常量

###### 数据流

使用一对@来定义一个数据流常量类型

例如

@FF0102030405060708090a0b0c0f@

数据流中,每字节数据都由成对的十六进制符定义完成,这代表序列中的数据必须被严格的配对定义

##### 变量类型

PainterScript支持以下几种类型

###### int

在PainterScript中,int类型被定义为一个32位的有符号整数.

###### float

在PainterScript中,float类型被定义为一个32位的有符号浮点数,该类型需要在符合IEEE754标准的编译器中被定义,否者,它将使用Q0.15的定点数标准进行实现..

###### string

字符串类型

###### memory

数据流类型

###### set

用户自定义集合类型

**在PainterScript中,所有变量必须预声明后使用,一个变量名在其有效域范围内仅能够被定义一次**

其BNF描述为

VARIABLE\_TYPE::=int|float|string|memory

VARIABLE\_DEFINE\_STATEMENT::=< VARIABLE\_TYPE > {<vName>[,]}

##### 变量定义

###### 常规变量定义

如下皆为合法常规变量定义

定义一个整形

int I;

连续定义多个整形

int a,b,c;

定义集合类型和集合数组

set a,b[10];

###### 指针类型

PainterScript支持一级指针,所有的变量类型皆可定义为一个一级指针类型

POINTER\_STATEMENT::=<VARIABLE\_TYPE> \* <vName>;

eg:如下代码定义了一个int类型指针

int \*p;

###### 数组

PainterScript支持一维数组,其BNF文法描述为

ARRAY\_STATEMENT::= <VARIABLE\_TYPE> <vName>”[“<Numeric>”]”;

eg:使用如下方式定义一个int类型数组

int a[10];

定义一个一级整形指针数组,和一个整形数组

int \*a[10],b[100];

###### 集合

PainterScript是参照C语言设计的一门脚本语言,在PainterScript中,使用数学描述Set(集合)来替代C语言的struct,同时,PainterScript吸取了C++中结构体的定义方式,定义一个集合文法如下

SET\_STATEMENT::= set <vName> “{” <….> ”}”

例如,下面是一个合法的定义例子

set MySet

{

int a,b,c;

float d;

string e[2];

memory f;

}

结构体运行相互嵌套,例如,下面的结构体定义也是合法的

set MySet2

{

MySet a,b[5];

int c;

}

##### 表达式

在PainterScript中,脚本由逻辑结构及表达式组成:

###### 运算符及优先级表

PainterScript参照了C语言的运算符进行设计,支持以下运算符

其中有所区别的是,sizeof被解析为返回内存类型或字符串类型的长度

|  |  |  |
| --- | --- | --- |
| 运算符 | 解释 | 结合方式 |
| () [] -> . | 括号（函数等），数组，两种结构成员访问 | 由左向右 |
| ! ~ ++ -- + -  \* & (类型) | 否定，按位否定，增量，减量，正负号，  间接，取地址，类型转换 | 由右向左 |
| \* / % | 乘，除，取模 | 由左向右 |
| + - | 加，减 | 由左向右 |
| << >> | 左移，右移 | 由左向右 |
| < <= >= > | 小于，小于等于，大于等于，大于 | 由左向右 |
| == != | 等于，不等于 | 由左向右 |
| & | 按位与 | 由左向右 |
| ^ | 按位异或 | 由左向右 |
| | | 按位或 | 由左向右 |
| && | 逻辑与 | 由左向右 |
| || | 逻辑或 | 由左向右 |
| = | 赋值运算 | 由右向左 |
| , | 逗号（顺序） | 由左向右 |

其运算符优先级同样参照C语言运算符的优先级表进行设计

1. ()  []  .  ->
2. !  ~   -（负号） ++  --   & \*

3   \* / %

4   + -

5   >> <<

6   > >= < <=

7   == !=

8   &

9   ^

10  |

11  &&

12  ||

13  ?

14   =

15  ,

###### 类型访问

int,float 类型访问

int及float变量类型可以直接对其助记符进行访问

例如

int a;

a=123;

a=a+3;

a++;

都是合法的访问

int 及 float的相互赋值将导致隐式转换,例如

int a;

a=3.14;

那么a的值将转换为3

a=5;

a=a/2;a的值为2;

string类型访问

string类型仅允许string类型及字符串常量间的相互操作

例如下面都是合法的语句

string foo;

foo=”abc”;

string类型允许加法运算

例如

string foo;

foo=”Hello”;

foo=foo+”World”;

foo=foo+foo;

//foo最终为HelloWorldHelloWorld;

同时,string类型允许访问其元素,例如

string foo=”HelloWolrd”;

int a;

a=foo[0];

//a的值为H的ase码

foo[0]=’M’;

//foo变换为MelloWorld;

memory类型访问

memory类型仅允许memory类型及字符串常量间的相互操作

例如下面都是合法的语句

memoryfoo;

foo=@010203@

memory类型允许加法运算,但加法仅接受一个范围为0x00-0xff的int类型,如果大于0xff的值将被直接截断

例如

memory foo;

foo=foo+0xff;

foo=foo+foo;

//foo最终为0xffff;

同时, memory类型允许访问其元素,例如

memoryfoo=@0102@;

int a;

a=foo[0];

//a的值为1

foo[0]=0xff;

//foo变换为ff 02;

set类型访问

直接访问set类型元素使用符合.,访问指针类型set使用->运算符

例如定义下列集合

set mySet

{

int a,b;

}

可以进行如下访问

mySet theSet;

theSet.a=99;

mySet \*pSet=&thenSet;

pSet->b=99;

###### 关键字类型

1. int(<Expression>) 转换为int类型,参数可以是一个整形,浮点,字符串类型
2. float(<Expression>)转换为float类型, 参数可以是一个整形,浮点,字符串类型
3. string(<Expression>)转换为string类型, 参数可以是一个整形,浮点,字符串,或内存类型
4. memory(<Expression>)转换为memory类型,参数可以是一个字符串或内存类型;
5. strlen(<Expression>) 内建函数,返回string的长度
6. memlen(<Expression>) 内建函数,返回memory的长度
7. sin(<Expression>) 内建函数,sin三角函数
8. cos(<Expression>) 内建函数,cos三角函数
9. break 跳出当前的循环结构,可以是一个while for compare结构
10. return <Expression> 函数返回值

###### 函数调用

PainterScript的函数调用方式为

FUNCTIONCALL\_STATEMENT::=<Function>(<…..>);

例如下面是一个合法的函数调用

add(1,2);

##### 语法结构

###### if条件判断语句

if语句的设计与C语言保持一致

IF\_STATEMENT::=if(<Expression>)

{<….>}

else

{

<…..>

}

或者是

if(<Expression>)

<Expression>

else

<Expression>

当if判断块中的Expression计算结果不为0时,执行对于语句块,否者跳转到else中执行,例如,下面是一个合法的if语句

if(a==1)

{

print(“a is 1”);

}

else

print (“a is not 1”);

###### while循环语句

while语句的设计与C语言保持一致

WHILE\_STATEMENT::=while(<Expression>)

{<….>}

或者是

while(<Expression>)

<Expression>

当while判断块中的Expression计算结果不为0时,循环执行器对应语句块,例如下面是合法的while语句

while(1)

{

//do something

}

###### for循环语句

for语句的设计与C语言保持一致

FOR\_STATEMENT::=for(<init>;<condition>;<extend>)

{<….>}

或者是

for(<init>;<condition>;<extend>)

<Expression>

其中,init表达式会先执行一次,然后判断condition表达式是否为1,如果是,执行对于语句块,否者跳出for循环,最后执行extend,再次判断condition

###### compare语句

compare语句是PainterScript中独有的语法结构其语法格式为

compare(<VARIABLE>)

{

with <Expression1;Expression2;….>

{

<……>

}

}

即VARIABLE将会与with对于的表达式进行比较,如果满足with语句中任意表达式对等,则执行with语句块中的语句,compare语句设计用于简化if-elseif-else..结构,并设计为switch语句的替代语句.

下面的compare语句都是合法的

int a;

compare(a) with(1;2;3) print(“a is 1 or 2 or 3.”);

compare(a)

{

with(1)

{

print(“a is 1”);

}

with(2;1+2)

{

print(“a is 2 or 3”);

}

print(“compare done”);

}

##### 函数定义

PainterScript的函数定义与C语言基本一致

FUNCTION\_STATEMENT::=<VARIABLE\_TYPE> <FunctionName>(<….>)

在PainterScript中,函数必须被前置声明后使用,同时在PainterScript中,函数返回值不允许使用Set类型,其参数也不允许使用set类型,如果需要使用set类型引用,必须使用set指针进行传递,使用set指针作为返回值是合法的但需要注意局部set在函数执行结束后其占用的资源将被释放,这将应用该set将导致访问不可预知的数据.

PainterScript中,函数分三种类型(内定函数,导入函数(远程函数),导出函数),并使用两个关键字进行修饰,如果不使用修饰关键字,该函数将被当做是一个内定函数，作为一个内定函数，其在PainterScript中必须被实现

例如,下面是一个合法的内定函数定义

int MyFunction(int a,float b,string c,MySet \*pSet);

内定函数仅允许PainterScript脚本内部进行调用,虚拟机无法找到内定函数,如果你希望虚拟机可以调用该函数,它必须被声明为导出函数;

导入函数(远程函数)即以host进行修饰的函数,例如,下面是一个导入函数的声明

host void print(string msg);

导入函数即虚拟机中实现的函数,所有的导入函数都不能在PainterScript编写实现,例如

host void print(string msg)

{

//do something

}

是不合法的,导入函数的名称必须与虚拟机中注册的名称保持一致,如果调用一个不存在的导入函数,将会导致虚拟机报错.

导出函数即以export进行修饰的函数,导入函数可以在PainterScript中被内部调用或者被虚拟机中调用,例如,下面是一个导入函数的声明

export void main(string msg);

如果你希望虚拟机可以调用该函数,那么它必须被声明为导出函数,否者它无法被虚拟机所发现调用。作为一个导出函数，其在PainterScript中必须被实现。

下面是一个函数的示范代码

int add(int a,int b)

{

return a+b;

}

export int ss\_main()

{

return add(1,2);

}

##### 内联汇编

PainterVM允许使用内联汇编,格式为

\_asm

{

//内联的汇编语句

}

在内联汇编中,允许对变量进行访问,例如

int I;

\_asm

{

Mov I,1;

}

是被允许的

### PainterScript 简化编译器

#### 初始化编译器

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_CompilerInitialize(px\_memorypool \*mp,PX\_Compiler \*compiler); |
| 说明 | **初始化编译器** |
| 参数 | mp 编译器部署内存池  compiler 编译器实例 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 添加源代码

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_CompilerAddSource(PX\_Compiler \*compiler,const px\_char script[]); |
| 说明 | **将一段源代码加入编译器中,这个函数可以多次调用以加入多个源代码** |
| 参数 | compiler 编译器实例  script 代码 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 编译源代码

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_CompilerCompile(PX\_Compiler \*compiler,px\_memory \*bin,const px\_char entryScript[]); |
| 说明 | **编译源代码** |
| 参数 | compiler 编译器实例  bin 输出的二进制指令流  entryScript 入口函数名 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 释放编译器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_CompilerFree(PX\_Compiler \*compiler); |
| 说明 | **释放编译器** |
| 参数 | compiler 编译器实例 |
| 返回值 | - |

### PainterScript Asm汇编语法

#### 标志

汇编引擎以如下标号作为跳转标志

Name:

其中,Name为合法变量助记符

可以对其前置FUNCTION关键字

FUNCTION Name:

其效果和Name:等价

如果需要对标志进行导出以方便其它语言进行调用,必须前缀EXPORT关键字

语法规则为

EXPORT FUNCTION Name:

#### 汇编指令集

##### mov

赋值指令,将op2的值赋值给op1

mov [reg,local,global],[num,string,reg,local,global]

·

##### add

加法指令,op1=op1+op2

add [reg,local,global],[num,reg,local,global]

·

##### sub

减法指令,op1=op1-op2

sub [reg,local,global],[num,reg,local,global]

·

##### neg

符号求反指令,op1=-op1

neg [reg,local,global]

##### div

除法指令,op1=op1/op2

div [reg,local,global],[num,reg,local,global]

·

##### mul

乘法指令,op1=op1\*op2

mul [reg,local,global],[num,reg,local,global]

·

##### mod

余数指令,两个操作数必须为整数 op1=op1%op2

mod [reg,local,global],[int,reg,local,global]

##### shl

左移位指令,两个操作数必须为整数 op1=op1<<op2

shl [reg,local,global],[int,reg,local,global]

##### shr

右移位指令,两个操作数必须为整数 op1=op1>>op2

shr [reg,local,global],[int,reg,local,global]

##### and

与运算指令,op1=op1&op2

and [reg,local,global],[num,reg,local,global]

##### or

或运算指令,op1=op1|op2

or [reg,local,global],[num,reg,local,global]

·

##### xor

异或运算指令op1=op1^op2

xor [reg,local,global],[num,reg,local,global]

·

##### inv

取反指令,op1=~op1

inv [reg,local,global]

##### not

逻辑非指令 op1=!op1

not [reg,local,global]

##### andl

逻辑与指令 op1=op1&&op2

andl [reg,local,global],[num,reg,local,global]

##### orl

逻辑或指令 op1=op1||op2

andl [reg,local,global],[num,reg,local,global]

##### pow

阶乘指令(op1为底数,op2为指数,结果在op1中) op1=op1\_op2

pow [reg,local,global],[num,reg,local,global]

##### sin

正弦函数op1=sin(op2)

sin [reg,local,global],[num,reg,local,global]

##### cos

余弦函数 op1=cos(op2)

cos [reg,local,global],[num,reg,local,global]

##### int

强制类型转换为int型(原类型float)

int [reg,local,global]

##### flt

强制类型转换为float型

flt [reg,local,global]

##### strlen

字符型长度指令

op1=strlen(op2)

strlen [reg,local,global],[reg,local,global,string]

##### strcat

字符型拼接指令

strcat(op1,op2)

strcat [reg,local,global],[int,reg,local,global,string]

##### strrep

字符串替换函数

将op1存在的op2字符串替换为op3中的字符串, 注意:op2 op3必须为字符串类型

strrep [reg,local,global],[reg,local,global,string],[reg,local,global,string]

##### strchr

将op2在索引op3中的字存储在op1中, 注意:op2必须为字符串类型

strchr [reg,local,global],[reg,local,global,string],[reg,local,global,int]

##### strtoi

将op2转换为整数保存在op1中,注意:op2必须为字符串类型

strtoi [reg,local,global],[reg,local,global,string]

##### strtof

将op2转换为浮点数保存在op1中,注意:op2必须为字符串类型

strtof [reg,local,global],[reg,local,global,string]

##### strfri

将op2整数类型转换为字符串类型保存在op1中

strfri [reg,local,global],[reg,local,global,int]

##### strfrf

将op2浮点类型转换为字符串类型保存在op1中

strfrf [reg,local,global],[reg,local,global,float]

##### strset

将op1所在字符串索引为op2 int的字符置换为op3

如果op3为一个int,则取asc码(第八位1字节),如果op3为一个字符串,则取第一个字母

strset [reg,local,global],[reg,local,global,int],[reg,local,global,string,int]

##### strfind

在op2字符串中查找op3子字符串,如果找到了op1等于其开始索引,否者等于-1

如果op3为一个int,则取asc码(第八位1字节),如果op3为一个字符串,则取第一个字母

strset [reg,local,global],[reg,local,global,string],[reg,local,global,string]

##### strtmem

将op1字符串类型转换为内存类型

strfrf [reg,local,global]

##### asc

将op2的第一个字母以asc码的形式

asc [reg,local,global],[reg,local,global,string]

##### membyte

将op3 内存类型对应op2索引复制到op1中,这个类型是一个int类型(小于256)

membyte [reg,local,global],[reg,local,global,int],[reg,local,global,memory]

##### memset

设置op1对应op2索引的内存为op3

memset [reg,local,global],[reg,local,global,int],[reg,local,global ,int]

##### memtrm

将op1内存进行裁剪,其中,op2为开始位置,op2为大小

memcpy [reg,local,global],[reg,local,global,int],[reg,local,global,memory]

##### memfind

查找op2对应于op3内存所在的索引位置,返回结果存储在op1中,如果没有找到,op1将会置为-1

memfind [reg,local,global],[reg,local,global,memory],[reg,local,global,memory]

##### memlen

将op2的内存长度存储在op1中

memlen [reg,local,global],[reg,local,global,memory]

##### memcat

将op2的内存拼接到op1的尾部

memcat [reg,local,global],[int,reg,local,global,memory]

##### memtstr

将op1内存类型转换为字符串类型,如果op1的内存结尾不为0,将会被强制置为0

memtstr [reg,local,global]

##### datacpy

复制虚拟机data数据,从地址op2到地址op1,长度为op3

datacpy [reg,local,global,int], [reg,local,global,int], [reg,local,global,int]

##### jmp

跳转指令 跳转到op1地址

jmp [reg,num,local,global,label]

##### je

条件跳转,当op1等于op2,跳转到op3

je [num,string,reg,local,global],[num,string,reg,local,global],[reg,int,local,global,label]

##### jne

条件跳转,当op1不等于op2,跳转到op3

jne [num,string,reg,local,global],[num,string,reg,local,global],[reg,int,local,global,label]

##### jl

条件跳转,当op1小于op2,跳转到op3

jl [num,string,reg,local,global],[num,string,reg,local,global],[reg,int,local,global,label]

##### jle

条件跳转,当op1小于等于op2,跳转到op3

jle [num,string,reg,local,global],[num,string,reg,local,global],[reg,int,local,global,label]

##### jg

条件跳转,当op1大于op2,跳转到op3

jg [num,string,reg,local,global],[num,string,reg,local,global],[reg,int,local,global,label]

##### jge

条件跳转,当op1大于等于op2,跳转到op3

jge [num,string,reg,local,global],[num,string,reg,local,global],[reg,int,local,global,label]

##### lge

逻辑比较指令,当op2等于op3时将op1置1,否则为0

lge [reg,local,global], [num,string,reg,local,global] , [num,string,reg,local,global]

##### lgne

逻辑比较指令,当op2等于op3时将op1置0,否则为1

lge [reg,local,global], [num,string,reg,local,global] , [num,string,reg,local,global]

##### lgz

逻辑比较指令,当op1等于0时将op1置1,否则为0

lgz [reg,local,global]

##### lggz

逻辑比较指令,当op1大于0时将op1置1,否则为0

lggz [reg,local,global]

##### lggez

逻辑比较指令,当op1大于等于0时将op1置1,否则为0

lggez [reg,local,global]

##### lglz

逻辑比较指令,当op1小于0时将op1置1,否则为0

lglz [reg,local,global]

##### lglez

逻辑比较指令,当op1小于等于0时将op1置1,否则为0

lglez [reg,local,global]

##### call

调用指令,如果op1是本地地址则将当期下一条指令地址压栈,然后跳转到op1,如果op1是一个host地址,则该call为一个hostcall,hostcall不会将返回地址压栈

call [reg,int,local,global,label,host]

\*Host Call的返回值在r[0]中

\*由被调用者清理堆栈

##### push

将op1压栈 sp-1,stack[0]=op1

push [num,reg,local,global,string,label]

##### pop

出栈，并将该值

pop [reg,local,global]

##### adr

取堆栈的绝对地址,返回该堆栈的绝对地址

ADR [reg,local,global], [local,global]

##### popn

将op1个元素出栈

popn [reg,local,global]

##### ret

返回,pop一个返回地址,跳转到该地址.

##### wait

等待一个信号量置为0,否者这个虚拟机实例将被暂时挂起(但并不不影响suspend标准位),在每个虚拟机实例中都有16个信号量,通过signal指令对这些信号量进行设置

##### signal

等待op1对应索引的信号量置为op2, 在每个虚拟机实例中都有16个信号量,这意味着op1的范围是0-15,当一个信号量被设置为非0值时,执行wait指令后改虚拟机实例会被阻塞,直到这个信号量被置为0时才能继续执行后续指令

##### bpx

如果启动了调试器,将会在该指令上断点,否者作为一个空指令

##### nop

空指令

#### 表达式

在文件开头以

.stack m

.global n

指定堆栈元素大小

注:以下参数表达reg表示寄存器,local表示局部变量,global表示全局变量,num表示数值,label表示标签或函数,host表示引擎API,int表示整数,float表示浮点数,string表示字符串类型

其中local表示对当前堆栈操作local[0]表示栈顶,local[1]表示栈顶往后的第一个指针

global 表示对当前堆操作,按照global[n] 表示，n为索引量

op1表示参数1,op2表示参数2….以此类推

在Label中以$标志做前缀,表示这是一个hostcall

{}包含符表示为一内存类型例如{02AFEFCCDDEE},两个字符为一字节,必须2字节对齐

“”包含符表示为一字符串类型,支持\n换行\t对齐转义

‘’表示这个字符代表的asc码值

### PainterScript Asm汇编器

#### 编译汇编代码

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptAsmCompile(px\_memorypool \*mp,px\_byte \*asmcode,px\_memory \*binmemory); |
| 说明 | **编译一个脚本汇编文件** |
| 参数 | mp 临时内存池  asmcode 以\0结束的汇编代码  binmemory 输出bin内存指针 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

### PainterVM虚拟机

#### vm实例

typedef struct \_\_PX\_ScriptVM\_Instance

{

px\_int VM\_memsize; //虚拟机使用元内存大小

px\_int stringsize; //字符串常量内存

px\_int memorysize; //二进制常量内存大小

px\_int binsize; //shellcode内存大小

px\_int funcCount; //导出函数数量

px\_int hostCount; //host函数数量

px\_bool Suspend; //是否被挂起

px\_int T; //当前执行线程索引

px\_byte \* \_bin; //shellcode指针

px\_byte \*\_memory; //二进制常量指针

px\_char \*\_string; //字符串常量指针

px\_memorypool \*mp; //归属内存池

PX\_SCRIPT\_EXPORT\_FUNCTION \*\_func;//导出函数表

PX\_SCRIPT\_ASM\_HOST\_NODE \*\_host;//hostcall表

PX\_SCRIPTVM\_VARIABLE \*\_mem;//元内存指针

px\_bool debug; //是否处于调试状态

px\_bool bp\_next;//下一条语句断点

px\_int bp\_IP;//断点指令地址

}PX\_ScriptVM\_Instance;

#### host函数定义

|  |  |
| --- | --- |
| 函数名 | typedef px\_bool (\*PX\_ScriptVM\_Function\_Modules)(PX\_ScriptVM\_Instance \*Ins,px\_void \*userptr); |
| 说明 | **PainterScript host函数声明标准** |
| 参数 | PX\_ScriptVM\_Instance 实例  userptr 用户指针 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |
| 备注 | 通过  #define PX\_ScriptVM\_STACK(Ins,i) ((Ins)->\_mem[(Ins)->BP+i])  访问传入参数,其中,第一个参数下标为0,第二个为1以此类推  通过  #define PX\_ScriptVM\_RETURN(Ins) ((Ins)->R[1])  设定host函数返回类型 |

**实例初始化**

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptVM\_InstanceInit(PX\_ScriptVM\_Instance \*Ins,px\_memorypool \*mp,px\_byte \*code,px\_int size); |
| 说明 | **初始化一个虚拟机实例** |
| 参数 | PX\_ScriptVM\_Instance 实例  mp 内存池  code 虚拟机指令流  size 指令流大小 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 执行虚拟机函数

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptVM\_InstanceRunFunction(PX\_ScriptVM\_Instance \*Ins,px\_int threadID, px\_char \*func,px\_int paramcount,...); |
| 说明 | **通过函数名执行脚本函数,这个函数被调用后,对应函数会立即执行直到其结束或被信号量中断** |
| 参数 | PX\_ScriptVM\_Instance 实例  thread 由thread索引线程执行该函数  func 函数名  paramcount 参数个数  … 由特定函数初始化的函数类型 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptVM\_InstanceRunFunctionIndex(PX\_ScriptVM\_Instance \*Ins,px\_int threadID, px\_int funcIndex,px\_int paramcount,...); |
| 说明 | **通过函数索引执行脚本函数, 这个函数被调用后,对应函数会立即执行直到其结束或被信号量中断** |
| 参数 | PX\_ScriptVM\_Instance 实例  thread 由thread索引线程执行该函数  funcIndex 函数索引  paramcount 参数个数  … 由特定函数初始化的函数类型 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptVM\_InstanceBeginThreadFunction(PX\_ScriptVM\_Instance \*Ins,px\_int threadID, px\_char \*func,px\_int paramcount,...); |
| 说明 | **通过函数名创建线程函数,这个函数被调用后,对应函数并不会立即执行,需要调用PX\_ScriptVM\_InstanceRunThread来执行对应线程** |
| 参数 | PX\_ScriptVM\_Instance 实例  thread 由thread索引线程执行该函数  func 函数名  paramcount 参数个数  … 由特定函数初始化的函数类型 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptVM\_InstanceBeginThreadFunctionIndex(PX\_ScriptVM\_Instance \*Ins,px\_int threadID, px\_int \*funcIndex,px\_int paramcount,...); |
| 说明 | **通过函数索引创建线程函数,这个函数被调用后,对应函数并不会立即执行,需要调用PX\_ScriptVM\_InstanceRunThread来执行对应线程** |
| 参数 | PX\_ScriptVM\_Instance 实例  thread 由thread索引线程执行该函数  funcIndex 函数索引  paramcount 参数个数  … 由特定函数初始化的函数类型 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 运行虚拟机

|  |  |
| --- | --- |
| 函数名 | PX\_SCRIPTVM\_RUNRETURN PX\_ScriptVM\_InstanceRunThread(PX\_ScriptVM\_Instance \*Ins ,px\_int tick); |
| 说明 | **执行虚拟机实例直到当前线程其指令片耗尽** |
| 参数 | PX\_ScriptVM\_Instance 实例  tick 指令片(执行指令数),如果这个值小于0,该线程将一直执行到线程退出或信号量中断 |
| 返回值 | PX\_SCRIPTVM\_RUNRETURN有以下几种返回方式  PX\_SCRIPTVM\_RUNRETURN\_ERROR =0,虚拟机执行发生运行时错误(非法指令?)  PX\_SCRIPTVM\_RUNRETURN\_TIMEOUT,时间片耗尽返回  PX\_SCRIPTVM\_RUNRETURN\_END,线程结束  PX\_SCRIPTVM\_RUNRETURN\_SUSPEND,线程被挂起  PX\_SCRIPTVM\_RUNRETURN\_WAIT,信号量中断返回 |

#### 线程上下文切换

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptVM\_InstanceThreadSwitch(PX\_ScriptVM\_Instance \*Ins,px\_int T); |
| 说明 | **切换线程上下文** |
| 参数 | PX\_ScriptVM\_Instance 实例  T 线程ID |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 终止线程

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ScriptVM\_ThreadStop(PX\_ScriptVM\_Instance \*Ins,px\_int ThreadId); |
| 说明 | **强制终止实例的一个线程,这个函数将在脚本函数最后ret时调用,它不会清除寄存器中的值.** |
| 参数 | PX\_ScriptVM\_Instance 实例  T 线程ID |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ScriptVM\_ThreadClear(PX\_ScriptVM\_Instance \*Ins,px\_int ThreadId); |
| 说明 | **清理一个线程中的堆栈和寄存器,调用该函数必须在该线程没有执行时调用** |
| 参数 | PX\_ScriptVM\_Instance 实例  T 线程ID |
| 返回值 | - |

#### 挂起线程

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ScriptVM\_ThreadSuspend (PX\_ScriptVM\_Instance \*Ins,px\_int ThreadId); |
| 说明 | **挂起实例的一个线程** |
| 参数 | PX\_ScriptVM\_Instance 实例  T 线程ID |
| 返回值 | - |

#### 恢复线程

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ScriptVM\_ThreadResume (PX\_ScriptVM\_Instance \*Ins,px\_int ThreadId); |
| 说明 | **恢复被挂起实例的一个线程** |
| 参数 | PX\_ScriptVM\_Instance 实例  T 线程ID |
| 返回值 | - |

#### 注册host函数

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptVM\_RegistHostFunction(PX\_ScriptVM\_Instance \*Ins,px\_char \*name,PX\_ScriptVM\_Function\_Modules funcModules,px\_void \*userptr); |
| 说明 | **注册一个host函数** |
| 参数 | PX\_ScriptVM\_Instance 实例  name 注册函数名  funcModules 函数指针  userptr 用户指针 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 释放虚拟机

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ScriptVM\_InstanceFree(PX\_ScriptVM\_Instance \*Ins); |
| 说明 | **释放一个虚拟机实例** |
| 参数 | PX\_ScriptVM\_Instance 实例 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 类型访问

#define PX\_ScriptVM\_STACK(Ins,i) 访问虚拟栈元

#define PX\_ScriptVM\_HOSTPARAM(Ins,i) 访问虚拟栈元,该宏用于Host访问hostcall函数调用参数,其中索引0表示第一个参数

#define PX\_ScriptVM\_LOCALPARAM(Ins,i) 访问虚拟栈元,该宏用于访问脚本内部函数调用参数,其中索引0表示第一个参数

#define PX\_ScriptVM\_GLOBAL(Ins,i) 访问虚拟堆元

#define PX\_ScriptVM\_RETURN(Ins) 访问寄存器R1

##### 弹出栈

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ScriptVM\_POPN(PX\_ScriptVM\_Instance \*Ins,px\_int n); |
| 说明 | **弹出n字节栈元** |
| 参数 | PX\_ScriptVM\_Instance 实例  n 栈元数目 |
| 返回值 | - |

##### host返回值

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ScriptVM\_RET(PX\_ScriptVM\_Instance \*Ins,PX\_SCRIPTVM\_VARIABLE ret); |
| 说明 | **设置R0寄存器(返回值寄存器),该函数主要用于hostcall中的返回值** |
| 参数 | PX\_ScriptVM\_Instance 实例  ret 值 |
| 返回值 | - |

##### 压栈

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ScriptVM\_PUSH(PX\_ScriptVM\_Instance \*Ins,PX\_SCRIPTVM\_VARIABLE val); |
| 说明 | **元数据压栈** |
| 参数 | PX\_ScriptVM\_Instance 实例  val 元数据 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | PX\_SCRIPTVM\_VARIABLE PX\_ScriptVM\_Varibale\_int(px\_int \_int);  PX\_SCRIPTVM\_VARIABLE PX\_ScriptVM\_Varibale\_float(px\_float \_float);  PX\_SCRIPTVM\_VARIABLE PX\_ScriptVM\_Varibale\_string(px\_string \_ref\_string);  PX\_SCRIPTVM\_VARIABLE PX\_ScriptVM\_Varibale\_memory(px\_memory \_ref\_memory);  PX\_SCRIPTVM\_VARIABLE PX\_ScriptVM\_Varibale\_const\_string(px\_char \*buffer);  PX\_SCRIPTVM\_VARIABLE PX\_ScriptVM\_Varibale\_const\_memory(px\_byte \*buffer,px\_int \_size); |
| 说明 | **构造对应类型元数据** |
| 参数 | ~ |
| 返回值 | 构造的元数据 |

#### 命令调试系统

PainterEngine PainterVM 提供一个基于命令的简单调试器,提供如下三个接口函数供使用

##### 1.启用调试器

px\_int PX\_ScriptVM\_DebuggerEnable(PX\_ScriptVM\_Instance \*Ins,PX\_ScriptVM\_DebuggerPrint \_printFunc,PX\_ScriptVM\_DebuggerCommand \_cmdFunc);

通过调用该函数来启用一个调试器,其中PX\_ScriptVM\_DebuggerPrint是一个指向printf函数的函数指针,用于输出调试信息

PX\_ScriptVM\_DebuggerCommand 指向一个类似于scanf的函数指针用于中断输入命令

##### 2.调试器命令

调试器支持以下命令

B IP断点,BP后接addr 例如 B 100

N 单步执行下一条指令

C 继续执行

U 反汇编代码,U后接接下来的反汇编指令数目 例如 U 1400

R 查看寄存器,例如R 1表示查看寄存器1的值

G 查看Global数据,例如G 1表示查看Global[1]的数据

L 查看local数据,例如L 1表示查看Local[1]数据,Local数据本质是Global[BP+oft]

### Obj静态3D模型加载器

#### 加载器初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_3D\_ObjectDataInitialize(px\_memorypool \*mp,PX\_3D\_ObjectData \*ObjectData); |
| 说明 | **加载器初始化** |
| 参数 | mp 内存池  ObjectData 加载器对象 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 解析Object数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_3D\_ObjectDataLoad(PX\_3D\_ObjectData \*ObjectData,const px\_byte \*data,px\_int size); |
| 说明 | **解析Obj模型数据** |
| 参数 | ObjectData 加载器对象  data Obj数据指针  size 数据长度 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 数据转换到RenderList

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_3D\_ObjectDataToRenderList(PX\_3D\_ObjectData \*ObjectData,PX\_3D\_RenderList \*renderList); |
| 说明 | **将数据转换到PainterEngine 3D可渲染的RenderList** |
| 参数 | ObjectData 加载器对象  renderList PX\_3D\_RenderList对象 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 释放加载器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_3D\_ObjectDataFree(PX\_3D\_ObjectData \*ObjectData); |
| 说明 | **释放加载器** |
| 参数 | ObjectData 加载器对象 |
| 返回值 |  |

### 2dx动画描述语言

2dx动画文件是PainterEngine专门为逐帧动画设计的动画文件,在有文件的文件系统中,其常常以.2dx作为后缀,同时2dx makechain必须运行在有文件系统的操作系统中

2dx动画文件是一个依据简单描述脚本经过编译而成的一个动画文件,下面是一个2dx脚本

texture "1.traw" f1

texture "2.traw" f2

texture "3.traw" f3

animation clock

loop -1

tag begin

frame f1

sleep 100

frame f2

sleep 90

frame f3

sleep 80

goto begin

end

#### 指令

2dx脚本支持以下指令

**Texture 纹理映射指令**,其负责将脚本存在目录下的纹理文件映射为一助记符,在脚本中就可以通过该助记符来使用该纹理,其格式为

Texture “文件路径” 助记符

**Frame 纹理显示指令**,将助记符映射的纹理显示到当前帧上,其格式为

Frame 助记符

**Sleep 延迟指令**,延迟一段时间,其单位是毫秒,其格式为

Sleep 毫秒时间数

**Tag 标记**,其不是一个指令,它将搭配Goto指令进行使用,其使用方式为

Tag 标记符

**Animation 标记**,其不是一个指令,它将搭配Goto指令进行使用,实际上他和tag是同一作用的不同写法,用法为

Animation 标记符

**Loop指令**,修改Loop寄存器的值,他配合Goto 指令进行使用,其用法为

Loop 值

**Goto条件跳转指令**,跳转到标号的指令执行,在执行前其会检查Loop寄存器,如果Loop寄存器的值不为0,则跳转,同时假如loop寄存器的值不为-1,其会将loop寄存器的值-1,格式为

Goto 标记符

**END 结束指令**,用于通知动画播放器动画已经播放结束,是一个特殊指令,用法为

END

#### 编译

要编译一起2dx动画文件,你需要一个2dx脚本及对应纹理,并且将该脚本与纹理放在同一个目录中(否者你需要使用相对路径重新定位其在文件系统中的具体位置)

PainterEngine\_2dxMakeTool是PainterEngine附带的2dx动画编译器工具配件,其可以在带文件系统的windows及Linux衍生版本中编译成可执行的二进制文件并用于2dx动画编译

其使用方法为

PainterEngine\_2dxMakeTool 脚本路径 编译输出路径

如果脚本正确它将会输出一个2dx动画文件,你可以使用painterEngine将它加载到引擎当中

(注意,因为该工具并不附带脚本错误提示器,所以你必须确保你的脚本编写正确,不过不用担心,2dx的脚本语言本身较为简单,并不会给你编写带来多大困扰).

### 2dx动画

#### 2dx library

##### 通过2dx数据创建一个动画库

需要创建一个2dx动画,首先要加载2dx动画数据,并通过该数据创建一个库文件

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AnimationLibraryCreateFromMemory(px\_memorypool \*mp,px\_animationlibrary \*panimationLib,px\_byte \*\_2dxBuffer,px\_uint size); |
| 说明 | **通过2dx数据创建一个动画库** |
| 参数 | mp内存池  panimationLib px\_animationlibrary 库指针  \_2dxBuffer 2dx数据  size 2dx数据的长度 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 释放动画库文件

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationLibraryFree(px\_animationlibrary \*panimationLib); |
| 说明 | **释放动画库文件** |
| 参数 | panimation |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 取得动画库图像宽度

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_AnimationLibraryGetFrameWidth(PX\_Animationlibrary \*panimationLib,px\_int frameIndex); |
| 说明 | **取得动画库图像宽度** |
| 参数 | panimationLib 动画库  frameIndex 帧索引 |
| 返回值 | 图像宽度 |

##### 取得动画库图像高度

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_AnimationLibraryGetFrameHeight(PX\_Animationlibrary \*panimationLib,px\_int frameIndex); |
| 说明 | **取得动画库图像高度** |
| 参数 | panimationLib 动画库  frameIndex 帧索引 |
| 返回值 | 图像高度 |

#### 2dx animation

2dx animation动画是2dx library的实例化,每一个2dx animation都有自己独立的动画实例

每一个2dx animation都必须连接到一个2dx animation,并且通过update进行其动画的帧更新,在此之后,它就能渲染到对应的surface实现动画的播放了

##### 创建一个动画

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AnimationCreate(px\_animation \*animation,px\_animationlibrary \*linker); |
| 说明 | **通过2dx库数据创建一个动画** |
| 参数 | animation 动画对象  linker px\_animationlibrary 连接的库指针 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 取得动画对应动画库中的动画类型数量

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_AnimationGetAnimationsCount(PX\_Animation \*animation); |
| 说明 | **取得动画对应动画库中的动画类型数量** |
| 参数 | animation 动画对象 |
| 返回值 | 动画对应动画库中的动画类型数量 |

##### 设置当前播放动画

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AnimationSetCurrentPlayAnimation(PX\_Animation \*animation,px\_int i); |
| 说明 | **设置当前播放动画** |
| 参数 | animation 动画对象  I 当前播放动画索引 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AnimationSetCurrentPlayAnimationByName(PX\_Animation \*animation,px\_char \*name); |
| 说明 | **设置当前播放动画** |
| 参数 | animation 动画对象  name 动画名 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

##### 动画是否结束

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AnimationIsEnd(PX\_Animation \*panimation); |
| 说明 | **当前播放动画是否已经结束** |
| 参数 | animation 动画对象 |
| 返回值 | 如果是返回PX\_TRUE,否者PX\_FALSE |

##### 取得动画当前纹理

|  |  |
| --- | --- |
| 函数名 | px\_texture \*PX\_AnimationGetCurrentTexture(PX\_Animation \*panimation); |
| 说明 | **取得动画当前纹理** |
| 参数 | animation 动画对象 |
| 返回值 | 如果成功返回纹理指针,否者返回PX\_NULL |

##### 释放2dx动画对象

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationFree(px\_animation \*animation); |
| 说明 | **释放2dx动画对象** |
| 参数 | animation 动画对象 |
| 返回值 | - |

##### 重置该2dx动画对象

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationReset(px\_animation \*animation); |
| 说明 | **重置该2dx动画对象** |
| 参数 | animation 动画对象 |
| 返回值 | - |

##### 更新该动画对象

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationUpdate(px\_animation \*panimation,px\_uint elpased); |
| 说明 | **更新该动画对象** |
| 参数 | animation 动画对象  elapsed 距离上一次更新经过的时间(毫秒级) |
| 返回值 | - |

##### 渲染动画对象

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationRender(px\_surface \*psurface,px\_animation \*animation, px\_point position,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend); |
| 说明 | **渲染动画对象** |
| 参数 | animation 动画对象  psurface 渲染表面  position 偏移量  refPoint 渲染参考点  blend blend属性,参考PX\_TEXTURERENDER\_BLEND定义 |
| 返回值 | - |

##### 以一个旋转角度渲染动画对象

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationRenderRotation(px\_surface \*psurface,px\_animation \*animation, px\_point position,px\_int angle,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend); |
| 说明 | **渲染动画对象** |
| 参数 | animation 动画对象  psurface 渲染表面  position 偏移量  angle 旋转角度  refPoint 渲染参考点  blend blend属性,参考PX\_TEXTURERENDER\_BLEND定义 |
| 返回值 | - |

##### 以一个比例和方向渲染动画对象

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationRender(px\_surface \*psurface,px\_animation \*animation, px\_point position,px\_float scale,px\_point direction,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend); |
| 说明 | **渲染动画对象** |
| 参数 | animation 动画对象  psurface 渲染表面  position 偏移量  scale 比例  direction 方向,默认方向为(1,0,0)即x方向  refPoint 渲染参考点  blend blend属性,参考PX\_TEXTURERENDER\_BLEND定义 |
| 返回值 | - |

##### 以一个向量方向渲染动画对象(初始方向是x方向)

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationRender\_vector(px\_surface \*psurface,px\_animation \*animation, px\_point position,px\_point directionPX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend); |
| 说明 | **以一个向量方向渲染动画对象(初始方向是x方向)** |
| 参数 | animation 动画对象  psurface 渲染表面  position 偏移量  direction 方向向量, 默认方向为(1,0,0)即x方向  refPoint 渲染参考点  blend blend属性,参考PX\_TEXTURERENDER\_BLEND定义 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_AnimationRenderEx(px\_surface \*psurface,px\_animation \*animation, px\_point position,PX\_ALIGN refPoint,PX\_TEXTURERENDER\_BLEND \*blend,px\_float scale,px\_float rotation); |
| 说明 | **渲染动画对象** |
| 参数 | animation 动画对象  psurface 渲染表面  position 偏移量  refPoint 渲染参考点  blend blend属性,参考PX\_TEXTURERENDER\_BLEND定义  scale 缩放比例  rotation 旋转角度 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_AnimationIsEnd(px\_animation \*panimation); |
| 说明 | **判断当前动画是否已经结束** |
| 参数 | panimation 动画对象 |
| 返回值 | - |

|  |  |
| --- | --- |
| 函数名 | px\_rect PX\_AnimationGetSize(px\_animation \*panimation) |
| 说明 | **取得动画的建议最大宽高** |
| 参数 | panimation 动画对象 |
| 返回值 | rect类型,其中x,y必定为0  width,height为建议宽度 |

### PainterEngine LiveFramework

LiveFramework是PainterEngine的类似于live2D的动画展示

#### 导入LiveFramework

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LiveFrameworkImport(px\_memorypool \*mp,PX\_LiveFramework \*plive,px\_void \*buffer,px\_int size); |
| 说明 | **导入LiveFramework** |
| 参数 | mp 构造内存池  plive LiveFramework实例化  buffer live数据  size 数据长度 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### LiveFramework动画播放

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LiveFrameworkPlayAnimation(PX\_LiveFramework \*plive,px\_int index); |
| 说明 | **暂停播放**LiveFramework |
| 参数 | plive LiveFramework实例化  index 动画索引 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LiveFrameworkPlayAnimationByName(PX\_LiveFramework \*plive,const px\_char name[]); |
| 说明 | **暂停播放**LiveFramework |
| 参数 | plive LiveFramework实例化  name 动画名 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LiveFrameworkPlay (PX\_LiveFramework \*plive); |
| 说明 | **继续播放**LiveFramework |
| 参数 | plive LiveFramework实例化 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### LiveFramework暂停播放

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LiveFrameworkPause(PX\_LiveFramework \*plive); |
| 说明 | **暂停播放**LiveFramework |
| 参数 | plive LiveFramework实例化 |
| 返回值 |  |

#### 重置LiveFramework

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LiveFrameworkReset(PX\_LiveFramework \*plive); |
| 说明 | **重置**LiveFramework |
| 参数 | plive LiveFramework实例化 |
| 返回值 |  |

#### 停止LiveFramework

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LiveFrameworkStop(PX\_LiveFramework \*plive); |
| 说明 | **重置**LiveFramework |
| 参数 | plive LiveFramework实例化 |
| 返回值 |  |

#### 渲染LiveFramework

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LiveFrameworkRender(px\_surface \*psurface,PX\_LiveFramework \*plive,px\_int x,px\_int y,PX\_ALIGN refPoint,px\_dword elpased); |
| 说明 | **渲染live** |
| 参数 | psurface 渲染表面  plive LiveFramework实例化  x,y 渲染坐标  refPoint 参考点  elapsed 距离上次渲染经过的毫秒时间 |
| 返回值 |  |

### PainterEngine Live

PainterEngine Live是PainterEngineLiveFramework的引用对象

#### 创建Live

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LiveCreate(px\_memorypool \*mp,PX\_LiveFramework \*pLiveFramework,PX\_Live \*pLive); |
| 说明 | **创建Live** |
| 参数 | mp 构造内存池  pLiveFramework引用的LiveFramework  pLive Live实例 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 释放Live

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LiveFree(PX\_Live \*pLive); |
| 说明 | **释放Live** |
| 参数 | pLive Live实例 |
| 返回值 |  |

#### Live动画播放

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LivePlayAnimation(PX\_Live \*plive,px\_int index); |
| 说明 | **暂停播放Live** |
| 参数 | plive live实例化  index 动画索引 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LivePlayAnimationByName(PX\_Live \*plive,const px\_char name[]); |
| 说明 | **暂停播放Live** |
| 参数 | plive live实例化  name 动画名 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_LivePlay (PX\_Live \*plive); |
| 说明 | **继续播放Live** |
| 参数 | plive live实例化 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### Live暂停播放

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LivePause(PX\_LiveFramework \*plive); |
| 说明 | **暂停播放Live** |
| 参数 | plive live实例化 |
| 返回值 |  |

#### 重置live

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LiveReset(PX\_LiveFramework \*plive); |
| 说明 | **重置live** |
| 参数 | plive live实例化 |
| 返回值 |  |

#### 停止live

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LiveStop(PX\_LiveFramework \*plive); |
| 说明 | **重置live** |
| 参数 | plive live实例化 |
| 返回值 |  |

#### 渲染live

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_LiveRender(px\_surface \*psurface,PX\_LiveFramework \*plive,px\_int x,px\_int y,PX\_ALIGN refPoint,px\_dword elpased); |
| 说明 | **渲染live** |
| 参数 | psurface 渲染表面  plive live实例化  x,y 渲染坐标  refPoint 参考点  elapsed 距离上次渲染经过的毫秒时间 |
| 返回值 |  |

### 粒子系统

#### 粒子发射器参数

|  |  |
| --- | --- |
| 发射器参数名 | 说明 |
| tex | **粒子的纹理** |
| position | **发射器的位置** |
| direction | **发射器发射方向** |
| deviation\_rangAngle | **发射方向的角度容差**  **(容差是指发射方向随机的+-容差角度范围)** |
| velocity | **粒子发射速度** |
| deviation\_velocity | **速度容差**  **(容差是指发射速度随机的+-容差范围)** |
| atomsize | **粒子比例默认为1.0比例** |
| deviation\_atomsize | **比例容差** |
| rotation | **粒子旋转角速度** |
| deviation\_rotation | **旋转角速度容差** |
| alpha | **粒子透明度** |
| deviation\_alpha | **透明度容差** |
| hdrR | **红色混合比,默认为1.0** |
| deviation\_hdrR | **红色混合比容差** |
| hdrG | **绿色混合比,默认为1.0** |
| deviation\_hdrG | **绿色混合比容差** |
| hdrB | **蓝色混合比,默认为1.0** |
| deviation\_hdrB | **蓝色混合比容差** |
| sizeincrease | **粒子每秒的自增比例,如果大于1.0粒子将不断增大,0-1.0则不断缩小**  **不能为负数** |
| alphaincrease | **粒子每秒的透明度变换,0-1.0粒子将会越来越透明,不能为负数** |
| a | **加速度向量** |
| ak | **阻尼系数,默认为1.0,如果大于1.0,速度将越来越快,小于1.0则越来越慢,不能为负数** |
| alive | **粒子的存在时间,毫秒为单位** |
| generateDuration | **粒子发射间隔,毫秒为单位,如果为0则一次性发射最大数量,不能为负数** |
| maxCount | **场上同时存在的最多粒子数量** |
| launchCount | **发射粒子数量,每发射一个粒子,这个值会-1,直到这个值为0不再发射,如果这个值一开始设置为一个小于0的值,那么,粒子将一直发射直到达到场上同时存在的最多粒子数量** |
| Create\_func | **自定义函数,如果这个函数指针不为空,当创建了一个粒子后,将调用这个函数** |
| Update\_func | **自定义函数,如果这个函数指针不为空,当更新了一个粒子物理信息后,将调用这个函数,可以对粒子的物理属性进一步修正** |

#### 以默认值初始化一个发射器参数

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ParticalLauncherInitializeDefault(PX\_ParticalLauncher\_InitializeInfo\* info); |
| 说明 | **以默认值初始化一个发射器参数** |
| 参数 | 要初始化的发射器参数 |
| 返回值 | - |

#### 创建粒子系统

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ParticalLauncherCreate(PX\_Partical\_Launcher \*launcher,px\_memorypool \*mp,PX\_ParticalLauncher\_InitializeInfo Info); |
| 说明 | **创建一个粒子发射器** |
| 参数 | launcher 发射器  mp 内存池  Info发射器参数 |
| 返回值 | - |

#### 渲染粒子发射器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ParticalRender(PX\_Partical\_Launcher \*launcher,px\_surface \*surface,px\_dword elpased); |
| 说明 | **渲染粒子系统** |
| 参数 | launcher 发射器  surface 渲染表面  elpased上一次渲染经过的毫秒时间 |
| 返回值 |  |

#### 释放粒子发射器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ParticalFree(PX\_Partical\_Launcher \*launcher); |
| 说明 | **释放一个粒子发射器** |
| 参数 | launcher 发射器 |
| 返回值 |  |

#### 设置粒子发射器的位置

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ParticalLauncherSetLauncherPosition(PX\_Partical\_Launcher \*launcher,px\_point position); |
| 说明 | **设置粒子系统的位置** |
| 参数 | launcher 发射器  direction 方向,这个方向是以局部坐标为准的 |
| 返回值 |  |

#### 设置粒子发射器喷射方向

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ParticalLauncherSetDirection(PX\_Partical\_Launcher \*launcher,px\_point direction) |
| 说明 | **设置粒子系统喷射方向** |
| 参数 | launcher 发射器  direction 方向,这个方向是以局部坐标为准的 |
| 返回值 |  |

### MQTT协议

MQTT for PainterEngine是基于MQTT3.1.1标准的PainterEngine内置支持协议,支持MQTT Client的基础操作,MQTT Broker(Server)可由其他供应商或公有库架设

#### 初始化协议

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MQTTInitialize(PX\_MQTT \*Mqtt,px\_memorypool \*mp,PX\_Linker \*linker); |
| 说明 | **初始化MQTT协议** |
| 参数 | mqtt 实例  mp 内存池  linker 通讯链接器 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### MQTT连接

|  |  |
| --- | --- |
| 函数名 | PX\_MQTT\_CONNECT PX\_MQTTConnect(PX\_MQTT \*Mqtt,PX\_MQTT\_ConnectDesc connectDesc); |
| 说明 | **使用MQTT协议进行连接** |
| 参数 | mqtt 实例  conndesc 连接描述结构体  typedef struct  {  const px\_char \*userName; 用户名  const px\_char \*password; 密码  const px\_char \*willTopic; 遗嘱主题  const px\_void \*willContent; 遗嘱内容  px\_uint willSize; 遗嘱内容长度  px\_bool willRetain; 遗嘱保留标志  px\_bool CleanSession; 清理session  PX\_MQTT\_QOS\_LEVEL willQoS; 遗嘱Qos级别  px\_word KeepAliveTime;//seconds 连接保留时间(秒)  }PX\_MQTT\_ConnectDesc; |
| 返回值 | PX\_MQTT\_CONNECT枚举类型  PX\_MQTT\_CONNECT\_SUCCEEDED 成功连接  PX\_MQTT\_CONNECT\_ERROR\_UNKNOW 未知错误  PX\_MQTT\_CONNECT\_ERROR\_DISCONNECT 链接器错误/无法连通  PX\_MQTT\_CONNECT\_ERROR\_PROTOCAL\_NO\_SUPPORT 协议不支持  PX\_MQTT\_CONNECT\_ERROR\_SESSION\_ILLEGAL session数据不合法  PX\_MQTT\_CONNECT\_ERROR\_SERVER\_CRASH 服务器错误  PX\_MQTT\_CONNECT\_ERROR\_USER\_WRONG 登录账户错误  PX\_MQTT\_CONNECT\_ERROR\_ILLEGAL 数据不合法 |

#### 发布数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MQTTPublish(PX\_MQTT \*Mqtt,PX\_MQTT\_PublishDesc publishDesc); |
| 说明 | **使用MQTT发布一个数据** |
| 参数 | mqtt 实例  publishDesc 发布描述  typedef struct  {  PX\_MQTT\_QOS\_LEVEL qosLevel; qos级别  px\_bool retain; 保留标志  const px\_char \*Topic; 主题  const px\_void \*payload; payload  px\_int payloadSize; payload大小  }PX\_MQTT\_PublishDesc; |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 订阅数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MQTTSubscribe(PX\_MQTT \*Mqtt,PX\_MQTT\_SubscribeDesc subscribeDesc); |
| 说明 | **使用MQTT订阅数据** |
| 参数 | mqtt 实例  subscribeDesc订阅描述  typedef struct  {  PX\_MQTT\_QOS\_LEVEL qosLevel; 最大支持Qos级别  const px\_char \*Topic; 订阅主题  }PX\_MQTT\_SubscribeDesc; |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE,注意,返回PX\_FALSE可能出现了不可挽回错误,可能需要修正该错误再使用该协议 |

#### 取消订阅数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MQTTUnsubscribe(PX\_MQTT \*Mqtt,const px\_char \*Topic); |
| 说明 | **取消订阅MQTT主题** |
| 参数 | mqtt 实例  Topic 主题名字 |
| 返回值 | - |

#### MQTT Ping

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MQTTPingReq(PX\_MQTT \*Mqtt); |
| 说明 | **Ping** |
| 参数 | Mqtt 实例 |
| 返回值 | - |

#### 监听

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_MQTTListen(PX\_MQTT \*Mqtt); |
| 说明 | **监听MQTT,直到收到了一个合法的Publish数据返回PX\_TRUE,否者可能因为超时返回PX\_FALSE**  **获取最后一次返回主题请访问Mqtt->Topic**  **获取最后一次返回内容请访问Mqtt->payload** |
| 参数 | Mqtt 实例 |
| 返回值 | -**收到了一个合法的Publish数据返回PX\_TRUE,因为超时返回PX\_FALSE** |

#### 释放

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_MQTTFree(PX\_MQTT \*Mqtt); |
| 说明 | **释放MQTT实例** |
| 参数 | Mqtt 实例 |
| 返回值 | - |

### 资源管理器

painterengine提供资源管理器,所有的资源从其它存储介质加载后将被以特定的格式保存在资源管理器中,资源管理器所有的对象均由一个唯一的key进行标识存储,其内部使用CRC+红黑树的形式进行索引.

资源包括以下几种类型:

typedef enum

{

PX\_RESOURCE\_TYPE\_NULL, //无类型

PX\_RESOURCE\_TYPE\_ANIMATIONLIBRARY,//动画集

PX\_RESOURCE\_TYPE\_SCRIPT,//编译脚本

PX\_RESOURCE\_TYPE\_TEXTURE,//纹理

PX\_RESOURCE\_TYPE\_SHAPE,//轮廓

PX\_RESOURCE\_TYPE\_DATA,//数据类型

}PX\_RESOURCE\_TYPE;

#### 初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ResourceLibraryInit(px\_memorypool \*mp,PX\_ResourceLibrary \*lib); |
| 说明 | **资源库初始化** |
| 参数 | mp 资源存放内存池  lib 资源库 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 加载一个资源

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ResourceLibraryLoad(px\_memorypool \*mptemp,PX\_ResourceLibrary \*lib,PX\_RESOURCE\_TYPE type,px\_byte \*data,px\_uint datasize,px\_char \*key); |
| 说明 | **加载资源** |
| 参数 | mptemp 用于资源剖析的临时内存池  lib 资源库  type 资源类型  data 数据缓存  datasize 数据大小  key 索引key |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 添加一个贴图资源

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ResourceLibraryAddTexture(PX\_ResourceLibrary \*lib,px\_char \*key,px\_texture \*pTexture); |
| 说明 | **加载贴图资源** |
| 参数 | lib 资源库  key 索引key  pTexture 需要加载的贴图,这个贴图数据将会被深拷贝到资源池中 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 读取资源

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ResourceLibraryGet(PX\_ResourceLibrary \*lib,PX\_Resource \*pRes,px\_char \*key); |
| 说明 | **读取对应key的资源类型** |
| 参数 | lib 资源库  pres 用于存储读取资源结构  key 对应字符索引 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 读取纹理资源

|  |  |
| --- | --- |
| 函数名 | px\_texture \*PX\_ResourceLibraryGetTexture(PX\_ResourceLibrary \*lib,px\_char \*key); |
| 说明 | **从资源中读取一个纹理** |
| 参数 | lib 资源库  key 对应字符索引 |
| 返回值 | 如果成功返回该资源指针,否者返回PX\_NULL |

#### 读取轮廓资源

|  |  |
| --- | --- |
| 函数名 | px\_shape \*PX\_ResourceLibraryGetShape(PX\_ResourceLibrary \*lib,px\_char \*key); |
| 说明 | **从资源中读取一个轮廓** |
| 参数 | lib 资源库  key 对应字符索引 |
| 返回值 | 如果成功返回该资源指针,否者返回PX\_NULL |

#### 读取动画资源

|  |  |
| --- | --- |
| 函数名 | px\_animationlibrary \*PX\_ResourceLibraryGetAnimationLibrary(PX\_ResourceLibrary \*lib,px\_char \*key); |
| 说明 | **从资源中读取一个动画集** |
| 参数 | lib 资源库  key 对应字符索引 |
| 返回值 | 如果成功返回该资源指针,否者返回PX\_NULL |

#### 读取脚本资源

|  |  |
| --- | --- |
| 函数名 | px\_animationlibrary \*PX\_ResourceLibraryGetScript(PX\_ResourceLibrary \*lib,px\_char \*key); |
| 说明 | **从资源中读取一个脚本资源** |
| 参数 | lib 资源库  key 对应字符索引 |
| 返回值 | 如果成功返回该资源指针,否者返回PX\_NULL |

#### 读取声音资源

|  |  |
| --- | --- |
| 函数名 | px\_sounddata \*PX\_ResourceLibraryGetSound(PX\_ResourceLibrary \*lib,px\_char \*key); |
| 说明 | **从资源中读取一个脚本资源** |
| 参数 | lib 资源库  key 对应字符索引 |
| 返回值 | 如果成功返回该资源指针,否者返回PX\_NULL |

#### 读取数据资源

|  |  |
| --- | --- |
| 函数名 | px\_memory \*PX\_ResourceLibraryGetData(PX\_ResourceLibrary \*lib,px\_char \*key); |
| 说明 | **从资源中读取一个脚本资源** |
| 参数 | lib 资源库  key 对应字符索引 |
| 返回值 | 如果成功返回该资源指针,否者返回PX\_NULL |

#### 删除资源

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_ResourceLibraryGet(PX\_ResourceLibrary \*lib, px\_char \*key); |
| 说明 | **删除对应key的资源类型** |
| 参数 | lib 资源库  key 对应字符索引 |
| 返回值 | - |

#### 释放资源库

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_ResourceLibraryGet(PX\_ResourceLibrary \*lib,PX\_Resource \*pRes,px\_char \*key); |
| 说明 | **读取对应key的资源类型** |
| 参数 | lib 资源库  pres 用于存储读取资源结构  key 对应字符索引 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

### 帧同步协议

painterEngine提供两种网络同步协议

一种是可用于多人游戏中帧同步或状态同步的开发,提供高容错,延迟优化,断线重连,数据平滑,第三方数据扰乱,数据攻击,等优化

在默认情况下,该同步协议的平滑延迟为66毫秒,约每秒15帧,在所有客户端连接后,服务端开始以这个频率对所有游戏客户端进行同步

一种是数据同步协议,该协议将保证每一个客户端的数据和服务端数据同步.

#### 同步服务端

##### 服务端初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SyncFrameServerInit(PX\_Sync\_Server \*sync,px\_memorypool \*mp,px\_dword updateDuration,PX\_Linker \*linker); |
| 说明 | **同步器服务端初始化** |
| 参数 | sync同步器结构体  mp 内存池  updateduration 平滑延迟  Linker 通讯链接器,可以使用任何不可靠通信IO实现,但是数据丢失率过高可能会导致该同步算法有高卡顿现象,可以通过提高PX\_SYNC\_CLIENT\_SEND\_DURATION和PX\_SYNC\_SERVER\_SEND\_TIMES减轻该丢包造成的延迟,但这也意味着将占有更大的带宽 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 服务端添加客户端

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SyncFrameServerAddClient(PX\_Sync\_Server \*sync,px\_dword verify\_id,px\_dword client\_verify\_id,px\_dword c\_id); |
| 说明 | **服务端添加客户端** |
| 参数 | sync同步器结构体  verify\_id 验证id,这个id需要以安全随机数产生不可预测,与client\_verify\_id成对使用  client\_verify\_id 客户端验证id, 这个id需要以安全随机数产生不可预测,与verify\_id成对使用,该表示是验证客户端的唯一标准方式  c\_id 用于客户端的玩家标识,将会随着指令一通写入到指令流当中 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 服务端继续运行

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SyncFrameServerRun(PX\_Sync\_Server \*sync); |
| 说明 | **服务端继续运行** |
| 参数 | sync同步器结构体 |
| 返回值 | - |

##### 服务端暂停运行

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SyncFrameServerStop(PX\_Sync\_Server \*sync); |
| 说明 | **服务端暂停运行** |
| 参数 | sync同步器结构体 |
| 返回值 | - |

##### 服务端更新循环

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SyncFrameServerUpdate(PX\_Sync\_Server \*sync,px\_dword elpased); |
| 说明 | **服务端更新** |
| 参数 | sync同步器结构体  elapsed 距离上次更新经过的时间,注意,该值应该做到尽可能的小,否者可能导致过大的抖动 |
| 返回值 | - |

##### 释放服务端

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SyncFrameServerFree(PX\_Sync\_Server \*sync); |
| 说明 | **释放服务端资源** |
| 参数 | sync同步器结构体 |
| 返回值 | - |

##### 取得数据累加和校验

|  |  |
| --- | --- |
| 函数名 | px\_uint32 PX\_SyncFrameServerSum32(PX\_Sync\_Server \*sync); |
| 说明 | **计算当前同步数据的累加和,注意,只有当一帧结束后进行累加校验才有参考意义** |
| 参数 | sync同步器结构体 |
| 返回值 | - |

#### 帧客户端

##### 客户端初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_SyncFrameClientInit(PX\_Sync\_Client \*client,px\_memorypool \*mp,px\_dword updateDuration,PX\_Sync\_Port serverport,px\_dword server\_verify\_id,px\_dword verify\_id, PX\_Linker \*linker); |
| 说明 | **同步器客户端初始化** |
| 参数 | client同步器结构体  mp 内存池  verify\_id 验证id,这个id需要以安全随机数产生不可预测,与server\_verify\_id成对使用  erver\_verify\_id 客户端验证id, 这个id需要以安全随机数产生不可预测,与verify\_id成对使用,该表示是验证客户端的唯一标准方式  Linker 通讯链接器,可以使用任何不可靠通信IO实现,但是数据丢失率过高可能会导致该同步算法有高卡顿现象,可以通过提高PX\_SYNC\_CLIENT\_SEND\_DURATION和PX\_SYNC\_SERVER\_SEND\_TIMES减轻该丢包造成的延迟,但这也意味着将占有更大的带宽 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 添加指令

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SyncFrameClientAddInstr(PX\_Sync\_Client \*client,px\_void \*instr,px\_int size); |
| 说明 | **添加客户端指令流,该指令流会被所有客户端同步,同步时间由服务端确定** |
| 参数 | client同步器结构体  instr 指令流指针  size 指令流长度 |
| 返回值 | - |

##### 更新客户端

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SyncFrameClientUpdate(PX\_Sync\_Client \*sync,px\_dword elpased); |
| 说明 | **更新客户端** |
| 参数 | client同步器结构体  elapsed 距离上次更新经过的时间,注意,该值应该做到尽可能的小,否者可能帧率不稳定 |
| 返回值 | - |

##### 释放客户端

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_SyncFrameClientFree(PX\_Sync\_Client \*sync); |
| 说明 | **释放客户端资源** |
| 参数 | sync同步器结构体 |
| 返回值 | - |

##### 取得数据累加和校验

|  |  |
| --- | --- |
| 函数名 | px\_uint32 PX\_SyncFrameClientSum32(PX\_Sync\_Client \*sync); |
| 说明 | **计算当前同步数据的累加和,注意,只有当一帧结束后进行累加校验才有参考意义** |
| 参数 | sync同步器结构体 |
| 返回值 | - |

### JSON

PainterEngine支持对Json格式的解析,但不支持字符串转义,如果需要转义请自行处理

typedef struct \_PX\_Json\_Object

{

px\_list values;//用于存储PX\_Json\_Value的链表

}PX\_Json\_Object;

typedef struct

{

PX\_JSON\_VALUE\_TYPE type;//值数据类型

px\_string name;//值名

union

{

px\_string \_string;//字符串数据

px\_double \_number;//数值数据

px\_bool \_boolean;//布尔类型数据

PX\_Json\_Object json\_Object;//对象数据

px\_list \_Array;//数组数据

};

}PX\_Json\_Value;

#### 初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonInitialize(px\_memorypool \*mp,PX\_Json \*pjson); |
| 说明 | **初始化一个JSON结构体** |
| 参数 | Mp 内存池  Pjson json结构体 |
| 返回值 | -如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 读取JSON Object的子数据

|  |  |
| --- | --- |
| 函数名 | PX\_Json\_Value \*PX\_JsonGetObjectValue(PX\_Json\_Value \*json\_value,const px\_char name[]); |
| 说明 | **读取JSON Object的子数据** |
| 参数 | json\_value Object类型的JSon数据  name Json值名 |
| 返回值 | 对象类型指针 |

#### 读取JSON数据

|  |  |
| --- | --- |
| 函数名 | PX\_Json\_Value \* PX\_JsonGetValue(PX\_Json \*json,const px\_char \_payload[]) |
| 说明 | **读取JSON数据** |
| 参数 | Pjson json结构体  payload 加载文本  payload示范  读取类成员 Human.lily.age  支持数组读取 human.score[10].point |
| 返回值 | -对象类型指针 |

#### 读取JSON Array数据

|  |  |
| --- | --- |
| 函数名 | PX\_Json\_Value \* PX\_JsonGetArrayValue(PX\_Json\_Value \*value,px\_int i); |
| 说明 | **读取JSONArray数据** |
| 参数 | Value array类型json值  I array访问索引 |
| 返回值 | -对象类型指针 |

#### 解析JSON数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonParse(PX\_Json \*pjson,const px\_char \*json\_content); |
| 说明 | **解析JSON数据** |
| 参数 | Pjson json结构体  Json\_content JSON文本 |
| 返回值 | -如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 建立JSON数据类型

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonBuild\_Value(PX\_Json\_Value \*pValue,px\_string \*\_out,px\_bool bArrayValue); |
| 说明 | **建立一个JSON数据类型，建立类型插入到JSon结构中以建立JSON结构（这个函数为内部调用）** |
| 参数 | pValue JSON数据  \_out 名称  bArrayValue 数组类型 |
| 返回值 | -如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 将JSON结构转换为文本

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonBuild(PX\_Json \*pjson,px\_string \*\_out); |
| 说明 | **解析JSON数据** |
| 参数 | Pjson json结构体  Json\_content JSON文本 |
| 返回值 | -如果成功返回PX\_TRUE,否者返回PX\_FALSE |

#### 删除JSON结构中一个数据

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_JsonDelete(PX\_Json \*pjson,const px\_char payload[]); |
| 说明 | **删除JSon结构中数据** |
| 参数 | Pjson json结构体  Payload payload，例如A.B.C或A.B[1] |
| 返回值 | - |

#### 释放JSON

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_JsonFree(PX\_Json \*pjson); |
| 说明 | **释放JSON结构数据** |
| 参数 | Pjson json结构体 |
| 返回值 |  |

#### JSON数据建立

##### 建立Object数据类型

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonCreateObjectValue(px\_memorypool \*mp,PX\_Json\_Value \*pValue,const px\_char name[]); |
| 说明 | **建立Object数据类型** |
| 参数 | Mp 内存池  pValue 需要建立数据类型的父对象  name 数据名 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 建立String数据类型

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonCreateStringValue(px\_memorypool \*mp,PX\_Json\_Value \*pValue,const px\_char name[],const px\_char text[]); |
| 说明 | **建立Object数据类型** |
| 参数 | Mp 内存池  pValue 需要建立数据类型的父对象  name 数据名  text 数据的string |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 建立Number数据类型

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonCreateNumberValue(px\_memorypool \*mp,PX\_Json\_Value \*pValue,const px\_char name[],px\_double value); |
| 说明 | **建立Number数据类型** |
| 参数 | Mp 内存池  pValue 需要建立数据类型的父对象  name 数据名  value数据的值 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 建立Boolean数据类型

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonCreateBooleanValue(px\_memorypool \*mp,PX\_Json\_Value \*pValue,const px\_char name[],px\_bool b); |
| 说明 | **建立Boolean数据类型** |
| 参数 | Mp 内存池  pValue 需要建立数据类型的父对象  name 数据名  Boolean 布尔值 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 建立Array数据类型

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonCreateArrayValue(px\_memorypool \*mp,PX\_Json\_Value \*pValue,const px\_char name[]); |
| 说明 | **建立Array数据类型** |
| 参数 | Mp 内存池  pValue 需要建立数据类型的父对象  name 数据名 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 插入String数据类型到JSON结构中

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonAddString(PX\_Json \*pjson,const px\_char parent\_payload[],const px\_char name[],const px\_char text[]); |
| 说明 | **插入String数据类型到JSON结构中** |
| 参数 | pJson 目标数据  payload 数据访问代码,例如访问对象A中的B则为A.B,如B为数组则为A.B[数组索引]  name 数据名  text 文本 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 插入Number数据类型到JSON结构中

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonAddNumber(PX\_Json \*pjson,const px\_char parent\_payload[],const px\_char name[],px\_double number); |
| 说明 | **插入Number数据类型到JSON结构中** |
| 参数 | pJson 目标数据  payload 数据访问代码,例如访问对象A中的B则为A.B,如B为数组则为A.B[数组索引]  name 数据名  number 插入的number数据 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 插入Boolean数据类型到JSON结构中

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonAddBoolean(PX\_Json \*pjson,const px\_char parent\_payload[],const px\_char name[],const px\_bool b); |
| 说明 | **插入Boolean数据类型到JSON结构中** |
| 参数 | pJson 目标数据  payload 数据访问代码,例如访问对象A中的B则为A.B,如B为数组则为A.B[数组索引]  name 数据名  b Boolean数据 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 插入Array数据类型到JSON结构中

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonAddArray(PX\_Json \*pjson,const px\_char parent\_payload[],const px\_char name[]); |
| 说明 | **插入一个空的Array数据类型到JSON结构中** |
| 参数 | pJson 目标数据  payload 数据访问代码,例如访问对象A中的B则为A.B,如B为数组则为A.B[数组索引]  name 数据名 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 插入Object数据类型到JSON结构中

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonAddObject(PX\_Json \*pjson,const px\_char parent\_payload[],const px\_char name[]); |
| 说明 | **插入一个空的Object数据类型到JSON结构中** |
| 参数 | pJson 目标数据  payload 数据访问代码,例如访问对象A中的B则为A.B,如B为数组则为A.B[数组索引]  name 数据名 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 修改String数据类型的数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonSetString(PX\_Json \*pjson,const px\_char payload[],const px\_char text[]); |
| 说明 | **修改一个string数据类型为目标值** |
| 参数 | Mp 内存池  payload 数据访问代码,例如访问对象A中的B则为A.B,如B为数组则为A.B[数组索引]  text 需要修改成的文本类型 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 修改Number数据类型的数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonSetNumber(PX\_Json \*pjson,const px\_char payload[],const px\_double number); |
| 说明 | **修改一个Number数据类型为目标值** |
| 参数 | Mp 内存池  payload 数据访问代码  text 需要修改成的文本类型 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 修改Boolean数据类型的数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonSetBoolean(PX\_Json \*pjson,const px\_char payload[],const px\_bool b); |
| 说明 | **修改一个Boolean数据类型为目标值** |
| 参数 | Mp 内存池  payload 数据访问代码  b 需要修改成的文本类型 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 向一个Object数据类型中插入数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonObjectAddValue(PX\_Json\_Value \*pObject,PX\_Json\_Value \*value); |
| 说明 | **向一个Object数据类型中插入数据** |
| 参数 | pObject 目标Object  value 需要插入的数据 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

##### 向一个Array数据类型中插入数据

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_JsonArrayAddValue(PX\_Json\_Value \*pArray,PX\_Json\_Value \*value); |
| 说明 | **向一个Array数据类型中插入数据** |
| 参数 | pArray 目标Array  value 需要插入的数据 |
| 返回值 | 如果成功返回PX\_TRUE,否者返回PX\_FALSE |

### PainterEngine 基本对象

#### 使用PainterEngine绘制对象

PainterEngine的对象是由PainterEngine图形渲染函数进一步封装的结果，在PainterEngine中，所有的显示物都是以对象的形式存在的，对象是PainterEngine显示的基本形式，通过对对象的派生而衍生出不同的功能类型。

#### PainterEngine对象机制

PainterEngine的对象是所有控件的基础类型，一个对象当中包含了以下几种类型

1. 对象的父对象
2. 对象的前一个兄弟对象
3. 对象的后一个兄弟对象
4. 对象的子对象

用下图表示对象之间的级联关系



依照图所示，每一个Object都拥有4个链接点，向上方向指向该对象的父对象，向下指向该对象的子对象，向左指向该对象的前一个兄弟对象，向右指向改对象的后一个兄弟对象，当方向上不存在这个对象指向的对应节点的时候，这个节点将会也必须被赋值为NULL表示空。也就是，一个对象的四个方向节点，要么有效存在，要么为NULL。

当父对象为NULL且只存在子对象的时候，我们称之为对“根对象”，

根对象不存在父对象，也不存在兄弟对象，例如下图当中，被红色标志的Object即为根对象

对象的级联关系中，我们将上下级联的对象称之为“父子对象”，如下图所示，Object1与Object2互为父子



Object1是Object2的父对象，Object2是Object1的子对象

同时可以从上图看到Object2拥有2个兄弟对象，分别是Object3与Object4，在广义上Object3与Object4也是Object1的子对象，但是，Object3与Object4的父向量并不指向Object1而指向了NULL，这也意味着无法通过Object3或Object4直接寻找到其父对象而必须经过Object2.这也就意味着Object2是对象的入口，在同一层级的对象中，有且仅有一个入口对象，在上图中可以看到，Object2是Object1子对象的入口对象，Object5是Object3子对象的入口对象

PainterEngine提供了以下几个函数，用于对象的操作与使用

**PX\_Object \*PX\_ObjectCreate(PX\_Object \*Parent,int x,int y,int Width,int Height);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectCreate |
| 功能 | 创建一个对象 |
| 参数Parent | 创建对象的父对象 |
| 参数x | 对象的左上角坐标x |
| 参数y | 对象的左上角坐标y |
| 参数width | 对象的宽度 |
| 参数height | 对象的高度 |
| 返回值PX\_ObjectCreate \* | 创建对象的指针 |

当Parent为NULL的时候，表示这个对象是一个根对象，当对象不为NULL的时候，这个对象创建如下

创建对象的连接规则

1.当父对象没有子对象的时候，创建的对象将作为其子对象



2.当父对象拥有子对象的时候，这个对象将被插入到父对象的子对象层次的最后一个节点



**void PX\_ObjectInit(PX\_Object \*Object,PX\_Object \*Parent,int x,int y,int Width,int Height);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectInit |
| 功能 | 初始换一个对象 |
| 参数Object | 需要初始化的对象 |
| 参数Parent | 初始化该对象的父对象 |
| 参数x | 对象的左上角坐标x |
| 参数y | 对象的左上角坐标y |
| 参数width | 对象的宽度 |
| 参数height | 对象的高度 |

这个函数用于初始化自定义的一个对象，并将对象连接到父对象当中，连接的规则和创建对象的连接规则一样。

**void PX\_ObjectDelete(PX\_Object \*pObject);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectDelete |
| 功能 | 删除一个对象 |
| 参数Object | 指向该对象的指针 |

使用该函数将会将这个对象断开原有的对象链并删除，同时，这个对象所有的子对象都会被删除

如下图所示，删除Object将导致如下对象都被删除

删除完成后，Object2的下一个兄弟向量将被重新定位连接到Object4



如果删除的是Object2，则Object1的子对象将被重新定位到Object3，而Object3将作为Object1子对象的入口对象



**void PX\_DeleteLinkerObject(PX\_Object \*\*ppObject);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectLinkerDelete |
| 功能 | 删除该对象和该对象后续的所有兄弟对象 |
| 参数Object | 指向该对象的二级指针 |

**void PX\_ObjectSetPosition(PX\_Object \*Object,int x,int y);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectSetPosition |
| 功能 | 设置对象的位置 |
| 参数Object | 该对象指针 |
| 参数x | 对象的新横坐标x |
| 参数y | 对象的新纵坐标y |

**void PX\_ObjectSetSize(PX\_Object \*Object,int Width,int Height);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectSetSize |
| 功能 | 设置对象的大小 |
| 参数Object | 该对象的指针 |
| 参数width | 设置对象的新宽度 |
| 参数height | 设置对象的新高度 |

**void PX\_ObjectFree(PX\_Object \*Object);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectSetVisible |
| 功能 | 设置该对象可见性（将影响其子对象） |
| 参数Object | 该对象指针 |
| 参数Visible | 可见性，不为0表示可见，为0不可见 |

**void PX\_ObjectSetFocus(PX\_Object \*Object);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectSetFocus |
| 功能 | 设置该对象为焦点对象 |
| 参数Object | 该对象指针 |
| 备注 | 焦点对象会独占该对象树结构中的消息处理事件直到该对象的焦点属性被清除,通过设置LostFocusReleaseEvent,可销毁失去焦点事件,注意,清除焦点对象将会一并清除该对象所有父级对象的焦点属性 |

**void PX\_ObjectClearFocus(PX\_Object \*Object);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectClearFocus |
| 功能 | 取消对象的焦点属性 |
| 参数Object | 该对象指针 |
| 备注 | 焦点对象会独占该对象树结构中的消息处理事件直到该对象的焦点属性被清除,通过设置LostFocusReleaseEvent,可销毁失去焦点事件,注意,如果该对象的子对象中仍然有焦点对象,那么该对象的焦点标志不会被清除 |

#### 创建对象

非根对象是除根对象的附属对象，就如同对象机制中描述的那样，非根对象必须被链接到一个根对象当中，同时，可以进一步指派该对象的长宽高坐标位置等具体属性类型。

**PX\_Object \*PX\_ObjectCreate(px\_memorypool \*mp,PX\_Object \*Parent,px\_float x,px\_float y,px\_float z,px\_float Width,px\_float Height,px\_float Lenght);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectCreate |
| 功能 | 创建一个对象 |
| mp | 对象使用的内存池 |
| parent | 父对象(如果为NULL表示这是一个根对象) |
| x,y,z | 坐标信息,注意z坐标信息,z的值越大,这个对象在渲染时将被最先绘制,这也就意味着z值小的对象将会覆盖在z值较大的对象中 |
| Width,Heigth,Lenght | 宽度,高度,长度 |
| 返回值 | 返回创建的对象指针 |

注意,其它的非根对象需要被链接到一个根对象中,如果该对象是一个实际绘制对象,可以在创建完根对象后对其的具体属性进行进一步的赋值操作。

#### 设置对象的usercode

**px\_void PX\_ObjectSetUserCode(PX\_Object \*pObject,px\_int user);**

**px\_void PX\_ObjectSetUserPointer(PX\_Object \*pObject,px\_void \*user\_ptr);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectSetUserCode  PX\_ObjectSetUserPointer |
| 功能 | 设置对象的user字段,这个字段为用户自定义的类型,可以将这个值赋值为需要的数据类型,在对象更新渲染或者是事件回调时,利用该函数获取额外信息 |
| Object | 需要设置的对象指针 |
| user\_int  user\_ptr | 设置的值,注意,每个对象的user\_int和User\_ptr为共享内存的数据类型.建议仅能设置其中一个 |

#### 删除对象

**px\_void PX\_ObjectDelete(PX\_Object \*pObject);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectDelete |
| 功能 | 删除一个对象,注意,当需要删除对象的时候不能够使用MP\_Free函数,必须调用该函数来删除一个对象,该函数将会断开对象与父对象与兄弟对象的链接,整理对象关系 |
| Object | 需要删除的对象指针 |

**px\_void PX\_ObjectDeleteChilds( PX\_Object \*pObject );**

|  |  |
| --- | --- |
| 函数名 | PX\_ ObjectDeleteChilds |
| 功能 | 删除一个对象的所有子对象 |
| Object | 需要删除子对象的对象指针 |

#### 对象属性设置

**px\_void PX\_ObjectSetPosition(PX\_Object \*Object,px\_float x,px\_float y);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectSetPosition |
| 功能 | 设置一个对象的坐标 |
| Object | 需要设置的对象指针 |
| x,y,z | 三维坐标 |

**px\_void PX\_ObjectSetSize(PX\_Object \*Object,px\_float Width,px\_float Height,px\_float Length);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectSetSize |
| 功能 | 设置一个对象的宽度和高度 |
| Object | 需要设置的对象指针 |
| Width,Height,Length | 宽,高,长 |

**px\_void PX\_ObjectSetVisible(PX\_Object \*Object,px\_bool visible);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectSetVisible |
| 功能 | 设置一个对象的可见性 |
| Object | 需要设置的对象指针 |
| visible | 可见性,如果为FALSE这个对象将不被渲染 |
|  |  |

**px\_float PX\_ObjectGetHeight(PX\_Object \*Object);**

**px\_float PX\_ObjectGetWidth(PX\_Object \*Object);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectGetHeight  PX\_ObjectGetWidth |
| 功能 | 取得对象的高宽 |
| Object | 对象指针 |
| 返回值 | 对象的高宽 |

#### PainterEngine更新/绘制流水线

PainterEngine的绘制流程按照

先父对象，后子对象，先左兄弟对象，后右兄弟对象的流程进行绘制

PainterEngine提供两个对象绘制函数进行对象的绘制，分别是

**void PX\_ObjectUpdate(struct \_PX\_Object \*,px\_uint elpased);**

|  |  |
| --- | --- |
| 函数名 | ObjectUpdate |
| 功能 | 更新该对象及该对象的所有子对象 |
| 参数Object | 需要更新的对象指针 |
| 参数 elapsed | 上一次更新经过的时间 |

**void PX\_ObjectRender(px\_surface \*psurface,struct \_PX\_Object \*,px\_uint elpased);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectRender |
| 功能 | 绘制该对象及该对象的所有子对象 |
| 参数 psurface | 将对象渲染到的目标表面 |
| 参数Object | 需要绘制的对象指针 |
| 参数 elapsed | 上一次更新经过的时间 |

依照下图所示更新/绘制的绘制流程如下



Start🡪Object1🡪Object2🡪Object3🡪object5🡪object6🡪Object7🡪Object4🡪End;

### 对象事件机制

#### 事件类型

在PainterEngine中,每个对象都可以为其注册触发事件,当使用PX\_ObjectPostEvent函数对一个对象进行事件投递后,该对象及其子对象都会收到该事件信息,如果这个对象有注册这个事件的处理函数,将会回调该处理函数.

注意,如果对象树结构中有一个焦点对象,那么该焦点对象将会独占所有的消息事件处理的优先权并决定事件是销毁还是继续投递

|  |  |
| --- | --- |
| 事件定义 | 事件响应 |
| PX\_OBJECT\_EVENT\_ANY | 任意事件 |
| PX\_OBJECT\_EVENT\_CURSORMOVE | 收到指针移动事件,例如鼠标移动触发 |
| PX\_OBJECT\_EVENT\_CURSORUP | 收到指针抬起事件, 例如鼠标左键点击抬起,或者触摸屏触摸抬起 |
| PX\_OBJECT\_EVENT\_CURSORRDOWN | 收到指针按下事件, 例如鼠标右键点击按下 |
| PX\_OBJECT\_EVENT\_CURSORDOWN | 收到指针按下事件, 例如鼠标左键点击按下,或者触摸按下 |
| PX\_OBJECT\_EVENT\_CURSORRUP | 收到指针右键抬起事件, 例如鼠标右键抬起 |
| PX\_OBJECT\_EVENT\_CURSOROVER | 指针移进时触发一次 |
| PX\_OBJECT\_EVENT\_CURSOROUT | 指针移出时触发一次 |
| PX\_OBJECT\_EVENT\_CURSORWHEEL | 指针滚动,例如鼠标中键滚动或者触摸屏滑动 |
| PX\_OBJECT\_EVENT\_CURSORCLICK | 指针单机,例如鼠标单击或触摸屏单击, 这需要按下抬起都在同一个位置 |
| PX\_OBJECT\_EVENT\_CURSORDRAG | 指针拖拽,例如鼠标按下拖动或者触摸屏滑动 |
| PX\_OBJECT\_EVENT\_STRING | 输入法输入了一个字符串 |
| PX\_OBJECT\_EVENT\_EXECUTE | 对象被触发,例如按钮被点击触发(按钮触发有别于Click,click指按下抬起都在同一个位置) |
| PX\_OBJECT\_EVENT\_VALUECHANGED | 对象值改变,例如sliderbar或者list选择改变 |
| PX\_OBJECT\_EVENT\_DRAGFILE | 外部拖入了一个文件进来 |
| PX\_OBJECT\_EVENT\_KEYDOWN | 键盘某个按键被按下 |
| PX\_OBJECT\_EVENT\_IMPACT | 两个对象两两碰撞(这个事件尽在PainterEngine World中生效) |
| PX\_OBJECT\_EVENT\_ONFOCUSCHANGED | 焦点改变 |
| PX\_OBJECT\_EVENT\_SCALE | 缩放事件 |
| PX\_OBJECT\_EVENT\_WINDOWRESIZE | 窗口的大小发生了变化 |
| PX\_OBJECT\_EVENT\_ONFOCUS | 取得焦点 |
| PX\_OBJECT\_EVENT\_LOSTFOCUS | 失去焦点 |
| PX\_OBJECT\_EVENT\_CANCEL | 对象被取消(例如explorer控件点击取消) |
| PX\_OBJECT\_EVENT\_CLOSE | 对象被关闭(例如widget被关闭) |

#### 响应处理函数

响应处理函数是对象接收到对应事件调用的回调函数

其定义为

**px\_void (\*ProcessFunc)(PX\_Object \*,PX\_Object\_Event e,px\_void \*ptr);**

|  |  |
| --- | --- |
| 函数名 | ProcessFunc |
| 功能 | 事件回调函数 |
| 参数Object | 响应对象指针 |
| 参数Event | 响应的事件类型 |
| 参数ptr | 用户传入的指针 |

#### 注册对象事件

**px\_int PX\_ObjectRegisterEvent(PX\_Object \*Object,px\_uint Event,px\_void (\*ProcessFunc)(PX\_Object \*,PX\_Object\_Event e),px\_void \*ptr);**

|  |  |
| --- | --- |
| 函数名 | PX\_ObjectRegisterEvent |
| 功能 | 为一个对象注册响应事件 |
| 参数Object | 需要绘制的对象指针 |
| 参数Event | 响应的事件类型 |
| ProcessFunc | 响应处理函数 |
| ptr | 用户指针 |

#### 派分事件

**px\_void PX\_ObjectPostEvent(PX\_Object \*pPost,PX\_Object\_Event Event);**

|  |  |
| --- | --- |
| 函数名 | ObjectPostEvent |
| 功能 | 派分事件,该事件将会被派分到该对象及其子对象当中,如果该对象注册了该事件,将会执行该事件的回调函数,如果派分的对象树中有焦点对象,焦点会独占所有的事件处理 |
| 参数Object | 派分的对象指针 |
| 参数Event | 响应的事件 |

#### 构造事件

|  |  |
| --- | --- |
| 函数原型 | PX\_Object\_Event PX\_OBJECT\_BUILD\_EVENT(px\_uint Event); |
| 功能 | 构造一个事件 |
| 参数Event | 事件类型 |
| 返回值 | 构造的事件 |

#### 构造字符串事件

|  |  |
| --- | --- |
| 函数原型 | PX\_Object\_Event PX\_OBJECT\_BUILD\_EVENT\_STRING(px\_uint Event,const px\_char \*content); |
| 功能 | 构造一个字符串事件 |
| 参数Event | 事件类型 |
| 参数 content | 需要传入的字符串 |
| 返回值 | 构造的事件 |

#### 指针事件偏移

|  |  |
| --- | --- |
| 函数原型 | PX\_Object\_Event PX\_Object\_Event\_CursorOffset(PX\_Object\_Event e,px\_point offset); |
| 功能 | 将一个指针类型的坐标偏移一个位置 |
| 参数e | 传入事件 |
| 参数 offset | 偏移量 |
| 返回值 | 构造的事件 |

#### 取得指针事件的X坐标

|  |  |
| --- | --- |
| 函数原型 | px\_float PX\_Object\_Event\_GetCursorX(PX\_Object\_Event e); |
| 功能 | 取得指针事件的X坐标 |
| 参数e | 传入事件 |
| 返回值 | X坐标 |

#### 取得指针事件的Y坐标

|  |  |
| --- | --- |
| 函数原型 | px\_float PX\_Object\_Event\_GetCursorY(PX\_Object\_Event e); |
| 功能 | 取得指针事件的Y坐标 |
| 参数e | 传入事件 |
| 返回值 | Y坐标 |

#### 取得指针事件的Z坐标

|  |  |
| --- | --- |
| 函数原型 | px\_float PX\_Object\_Event\_GetCursorZ(PX\_Object\_Event e); |
| 功能 | 取得指针事件的Z坐标 |
| 参数e | 传入事件 |
| 返回值 | Z坐标 |

#### 设置指针事件的X坐标

|  |  |
| --- | --- |
| 函数原型 | px\_void PX\_Object\_Event\_SetCursorX(PX\_Object\_Event \*e,px\_float x); |
| 功能 | 设置指针事件的X坐标 |
| 参数e | 需要修改的传入事件 |
| 参数x | 需要设置的值 |
| 返回值 |  |

#### 设置指针事件的Y坐标

|  |  |
| --- | --- |
| 函数原型 | px\_void PX\_Object\_Event\_SetCursorY(PX\_Object\_Event \*e,px\_float y); |
| 功能 | 设置指针事件的Y坐标 |
| 参数e | 需要修改的传入事件 |
| 参数y | 需要设置的值 |
| 返回值 |  |

#### 设置指针事件的X坐标

|  |  |
| --- | --- |
| 函数原型 | px\_void PX\_Object\_Event\_SetCursorX(PX\_Object\_Event \*e,px\_float z); |
| 功能 | 设置指针事件的Z坐标 |
| 参数e | 需要修改的传入事件 |
| 参数z | 需要设置的值 |
| 返回值 |  |

#### 取得键盘事件的输入码

|  |  |
| --- | --- |
| 函数原型 | px\_uint PX\_Object\_Event\_GetKeyDown(PX\_Object\_Event e); |
| 功能 | 取得键盘事件的输入码 |
| 参数e | 传入事件 |
| 返回值 | 键盘输入码 |

#### 取得输入法事件的输入字符串

|  |  |
| --- | --- |
| 函数原型 | px\_char\* PX\_Object\_Event\_GetStringPtr(PX\_Object\_Event e); |
| 功能 | 取得输入法事件的输入字符串 |
| 参数e | 传入事件 |
| 返回值 | 输入字符串指针 |

### PainterEngine UI对象

PainterEngine UI是基于PainterEngine Object开发的默认UI控件,包含按钮,文本,图片,进度条,拖动框,菜单,下拉框……等一系列平台无关的交互控件

#### 静态文本框

##### 创建文本框

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_LabelCreate(px\_memorypool \*mp,PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height,const px\_char \*Text,PX\_FontModule \*fm,px\_color Color); |
| 功能 | 创建一个静态文本框 |
| mp | 使用内存池 |
| Parent | 父对象 |
| x,y | 二维坐标xy |
| width,height | 宽度,高度 |
| text | 文本 |
| fm | 字模,如果为PX\_NULL表示默认字模 |
| Color | 文本颜色 |
| 返回值 | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_Label \* PX\_Object\_GetLabel(PX\_Object \*Object); |
| 功能 | 取得静态文本框对象数据 |
| Object | 对象指针 |
| 返回值 | 如果成功返回对象数据,否者返回PX\_NULL |

##### 取得静态文本框文本

|  |  |
| --- | --- |
| 函数名 | px\_char \* PX\_Object\_LabelGetText(PX\_Object \*Label); |
| 功能 | 取得静态文本框的文本 |
| Object | 对象指针 |
| 返回值 | 如果成功返回文本框文本,否者返回PX\_NULL |

##### 设置静态文本框文本

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_LabelSetText(PX\_Object \*pLabel,px\_char \*Text); |
| 功能 | 设置静态文本框的文本 |
| Object | 对象指针 |
| Text | 文本指针 |
| 返回值 | - |

##### 设置文本框背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_LabelSetBackgroundColor(PX\_Object \*pLabel,px\_color Color); |
| 功能 | 设置文本框的背景颜色 |
| Object | 对象指针 |
| color | 颜色 |
| 返回值 | - |

##### 设置文本框文本对齐方式

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_LabelSetAlign(PX\_Object \*pLabel,PX\_FONT\_ALIGN Align); |
| 功能 | 设置文本框的文本对齐方式 |
| Object | 对象指针 |
| color | 颜色 |
| 返回值 | - |

#### 进度条

##### 创建进度条

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_ProcessBarCreate(px\_memorypool \*mp,PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height); |
| 功能 | 创建进度条 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_ProcessBar \*PX\_Object\_GetProcessBar(PX\_Object \*Object); |
| 功能 | 取得进度条对象数据 |
| Object | 对象指针 |
| 返回值 | 如果成功返回对象数据指针,否者返回PX\_NULL |

##### 设置进度条颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ProcessBarSetColor(PX\_Object \*pProcessBar,px\_color Color); |
| 功能 | 设置进度条颜色 |
| Object | 对象指针 |
| color | 颜色 |
| 返回值 | - |

##### 设置进度条当前值

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ProcessBarSetValue(PX\_Object \*pProcessBar,px\_int Value); |
| 功能 | 设置进度条最大值 |
| Object | 对象指针 |
| value | 当前值 |
| 返回值 | - |

##### 设置进度条最大值

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ProcessBarSetMax(PX\_Object \*pProcessBar,px\_int Max); |
| 功能 | 设置进度条最大值 |
| Object | 对象指针 |
| max | 最大值 |
| 返回值 | - |

##### 取得进度条当前值

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_Object\_ProcessBarGetValue(PX\_Object \*pProcessBar); |
| 功能 | 取得进度条当前值 |
| Object | 对象指针 |
| 返回值 | 进度条当前值 |

#### 图片框

##### 创建图片框

|  |  |
| --- | --- |
| 函数名 | PX\_Object \*PX\_Object\_ImageCreate(px\_memorypool \*mp,PX\_Object \*Parent,px\_int x,px\_int y,px\_texture \*ptex); |
| 功能 | 创建图片框 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| pTex | 纹理指针 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_Image \*PX\_Object\_GetImage(PX\_Object \*Object); |
| 功能 | 取得图片框对象数据 |
| Object | 对象指针 |
| 返回值 | 如果成功返回对象数据指针,否者返回PX\_NULL |

#### 滑动框

##### 创建滑动框

|  |  |
| --- | --- |
| 函数名 | PX\_Object \*PX\_Object\_SliderBarCreate(px\_memorypool \*mp,PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height,enum PX\_OBJECT\_SLIDERBAR\_TYPE Type,enum PX\_OBJECT\_SLIDERBAR\_STYLE style); |
| 功能 | 创建滑动框 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| PX\_OBJECT\_SLIDERBAR\_TYPE | 滑动框类型(水平,垂直) |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_SliderBar \*PX\_Object\_GetSliderBar(PX\_Object \*Object); |
| 功能 | 取得滑动框对象数据 |
| Object | 对象指针 |
| 返回值 | 如果成功返回对象数据指针,否者返回PX\_NULL |

##### 设置滑动框当前值

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SliderBarSetValue(PX\_Object \*pSliderBar,px\_int Value); |
| 功能 | 设置滑动框当前值 |
| Object | 对象指针 |
| value | 当前值 |
| 返回值 | - |

##### 设置滑动框值域

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SliderBarSetRange(PX\_Object \*pSliderBar,px\_int Min,px\_int Max); |
| 功能 | 设置滑动框值域 |
| Object | 对象指针 |
| Min | 最小值 |
| Max | 最大值 |
| 返回值 | - |

##### 读取滑动框值

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_Object\_SliderBarGetValue(PX\_Object \*pSliderBar); |
| 功能 | 读取滑动框值 |
| Object | 对象指针 |
| 返回值 | 当前滑动框的值 |

##### 设置滑动框背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SliderBarSetBackgroundColor(PX\_Object \*pSliderBar,px\_color color); |
| 功能 | 设置滑动框背景颜色 |
| Object | 对象指针 |
| color | 背景颜色 |
| 返回值 | 当前滑动框的值 |

##### 设置滑动框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SliderBarSetColor(PX\_Object \*pSliderBar,px\_color color); |
| 功能 | 设置滑动框颜色 |
| Object | 对象指针 |
| color | 颜色 |
| 返回值 |  |

##### 设置滑动框按钮大小

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SliderBarSetSliderButtonLength(PX\_Object \*pSliderBar,px\_int length); |
| 功能 | 取得滑动框当前值 |
| Object | 对象指针 |
| length | 按钮长度 |
| 返回值 |  |

#### 按钮

##### 创建按钮

|  |  |
| --- | --- |
| 函数名 | PX\_Object \*PX\_Object\_PushButtonCreate(px\_memorypool \*mp,PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height,const px\_char \*Text,PX\_FontModule \*fontmodule); |
| 功能 | 创建按钮 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| text | 按钮文本 |
| fontmodule | 字模 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_PushButton \* PX\_Object\_GetPushButton( PX\_Object \*Object ); |
| 功能 | 取得按钮对象数据 |
| Object | 对象指针 |
| 返回值 | 如果成功返回对象数据指针,否者返回PX\_NULL |

##### 取得按钮文本

|  |  |
| --- | --- |
| 函数名 | px\_char \* PX\_Object\_PushButtonGetText( PX\_Object \*PushButton ); |
| 功能 | 取得按钮文本 |
| Object | 对象指针 |
| 返回值 | 返回文本指针 |

##### 设置按钮文本

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetText( PX\_Object \*pObject,const px\_char \*Text ); |
| 功能 | 设置按钮文本 |
| Object | 对象指针 |
| Text | 想要设置的文本 |
| 返回值 | - |

##### 设置按钮背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetBackgroundColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置按钮背景颜色 |
| Object | 对象指针 |
| color | 想要设置背景颜色 |
| 返回值 | - |

##### 设置按钮指针颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetCursorColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置按钮背景指针颜色(当鼠标移动到按钮上显示的背景颜色) |
| Object | 对象指针 |
| color | 想要设置指针颜色 |
| 返回值 | - |

##### 设置按钮风格

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetStyle(PX\_Object \*pObject,PX\_OBJECT\_PUSHBUTTON\_STYLE style); |
| 功能 | 设置按钮风格 |
| Object | 对象指针 |
| style | 想要设置的风格(矩形按钮,圆角按钮) |
| 返回值 | - |

##### 设置按钮按下颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetPushColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置按钮按下颜色 |
| Object | 对象指针 |
| color | 想要设置的按下颜色 |
| 返回值 | - |

##### 设置按钮边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetBorderColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置按钮边框颜色 |
| Object | 对象指针 |
| color | 想要设置的边框颜色 |
| 返回值 | - |

##### 设置按钮边框是否显示边框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetBorder( PX\_Object \*Object,px\_bool Border ); |
| 功能 | 设置按钮是否显示边框 |
| Object | 对象指针 |
| Border | PX\_TRUE表示显示,否者(PX\_FALSE)表示不显示 |
| 返回值 | - |

##### 设置按钮文本颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetTextColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置按钮文本颜色 |
| Object | 对象指针 |
| color | 想要设置的文本颜色 |
| 返回值 | - |

##### 设置按钮背景图片

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetTexture(PX\_Object \*pObject,px\_texture \*texture); |
| 功能 | 设置按钮背景图片(这个图片会居中显示) |
| Object | 对象指针 |
| texture | 纹理指针 |
| 返回值 | - |

##### 设置按钮背景轮廓

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_PushButtonSetShape(PX\_Object \*pObject,px\_shape \*pshape); |
| 功能 | 设置按钮背景轮廓(这个轮廓会居中显示) |
| Object | 对象指针 |
| pShape | 轮廓指针 |
| 返回值 | - |

#### 文本编辑框

##### 创建文本编辑框

|  |  |
| --- | --- |
| 函数名 | PX\_Object\* PX\_Object\_EditCreate(px\_memorypool \*mp, PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height,PX\_FontModule \*fontModule); |
| 功能 | 创建文本编辑框 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| fontmodule | 字模 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_Edit \* PX\_Object\_GetEdit( PX\_Object \*Object ); |
| 功能 | 取得文本编辑框对象数据 |
| Object | 对象指针 |
| 返回值 | 如果成功返回对象数据指针,否者返回PX\_NULL |

##### 取得文本编辑框文本

|  |  |
| --- | --- |
| 函数名 | px\_char \* PX\_Object\_EditGetText( PX\_Object \*pObject ); |
| 功能 | 取得文本编辑框当前文本 |
| Object | 对象指针 |
| 返回值 | 返回文本字符串指针 |

##### 取得文本编辑框最大(字节)长度

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetMaxTextLength(PX\_Object \*pObject,px\_int max\_length); |
| 功能 | 取得文本编辑框最大长度 |
| Object | 对象指针 |
| 返回值 | - |

##### 设置文本编辑框文本

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetText( PX\_Object \*pObject,const px\_char \*Text ); |
| 功能 | 取得文本编辑框文本 |
| Object | 对象指针 |
| Text | 文本 |
| 返回值 | - |

##### 追加文本编辑框文本

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditAppendText( PX\_Object \*pObject,const px\_char \*Text ); |
| 功能 | 追加文本编辑框文本 |
| Object | 对象指针 |
| Text | 文本 |
| 返回值 | - |

##### 激活文本编辑框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetFocus( PX\_Object \*pObject,px\_bool OnFocus); |
| 功能 | 激活文本编辑框 |
| Object | 对象指针 |
| OnFocus | PX\_TRUE表示激活,否者不激活 |
| 返回值 | - |

##### 文本编辑框密码模式

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetPasswordStyle( PX\_Object \*pObject,px\_uchar Enabled ); |
| 功能 | 文本编辑框密码模式 |
| Object | 对象指针 |
| Enabled | PX\_TRUE表示启用,否者不启用 |
| 返回值 | - |

##### 文本编辑框背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetBackgroundColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 文本编辑框背景颜色 |
| Object | 对象指针 |
| color | 背景颜色 |
| 返回值 | - |

##### 文本编辑框边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetBorderColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 文本编辑框边框颜色 |
| Object | 对象指针 |
| color | 边框颜色 |
| 返回值 | - |

##### 文本编辑框指针颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetCursorColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 文本编辑框指针颜色(鼠标移动到编辑框时的背景颜色) |
| Object | 对象指针 |
| color | 指针颜色 |
| 返回值 | - |

##### 文本编辑框文本颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetTextColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 文本编辑框文本颜色 |
| Object | 对象指针 |
| color | 文本颜色 |
| 返回值 | - |

##### 文本编辑框过滤器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetLimit(PX\_Object \*pObject,const px\_char \*Limit); |
| 功能 | 文本编辑框过滤器(仅允许输入Limit中包含的ANSI字符) |
| Object | 对象指针 |
| Limit | 过滤器字符串,包含所有允许输入的ANSI字符 |
| 返回值 | - |

##### 文本编辑框显示边框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditSetBorder( PX\_Object \*pObj,px\_bool Border ); |
| 功能 | 文本编辑框显示边框 |
| Object | 对象指针 |
| Border | PX\_TRUE表示显示,PX\_FALSE表示不显示 |
| 返回值 | - |

##### 自动换行

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_EditAutoNewLine(PX\_Object \*pObject,px\_bool b,px\_int AutoNewLineSpacing); |
| 功能 | 是否允许文本编辑框自动换行(当字符超过文本编辑框宽度的时候) |
| Object | 对象指针 |
| b | PX\_TRUE表示允许,PX\_FALSE表示不允许 |
| AutoNewLineSpacing | 距离最右边框多少像素距离时换行 |
| 返回值 | - |

#### 列表框

##### 创建列表框

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_ListCreate(px\_memorypool \*mp, PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height,px\_int ItemHeight,PX\_Object\_ListItemOnCreate \_CreateFunc); |
| 功能 | 创建列表框 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| ItemHeight | 每个列表条目的高度 |
| CreateFunc | 列表条目的创建函数  List自动创建了一个类型为PX\_Object\_ListItem的对象,  你需要在这个创建函数中,设定这个对象的更新/渲染/释放 函数,你可以通过这个对象类型的pData获取对应条目的信息数据 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_ListItem \* PX\_Object\_GetListItem( PX\_Object \*Object ); |
| 功能 | 取得列表框对象数据 |
| Object | 对象指针 |
| 返回值 | 如果成功返回对象数据指针,否者返回PX\_NULL |

##### 清空列表框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ListClear(PX\_Object \*pListObj); |
| 功能 | 清空列表框 |
| Object | 对象指针 |
| 返回值 | - |

##### 列表框添加一个条目数据

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ListAdd(PX\_Object \*pListObj,px\_void \*ptr); |
| 功能 | 列表框添加一个条目数据 |
| Object | 对象指针 |
| ptr | 条目数据的指针,这个条目数据必须是一个生存期有效的数据指针 |
| 返回值 | - |

##### 取得条目数据

|  |  |
| --- | --- |
| 函数名 | px\_void \*PX\_Object\_ListGetItem(PX\_Object \*pListObject,px\_int index); |
| 功能 | 取得条目数据 |
| Object | 对象指针 |
| index | 条目索引 |
| 返回值 | 条目数据指针 |

##### 删除条目(数据)

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ListRemoveItem(PX\_Object \*pListObject,px\_int index); |
| 功能 | 删除条目数据 |
| Object | 对象指针 |
| index | 条目索引 |
| 返回值 | - |

##### 设置背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ListSetBackgroundColor(PX\_Object \*pListObject,px\_color color); |
| 功能 | 设置背景颜色 |
| Object | 对象指针 |
| index | 要设置的背景颜色 |
| 返回值 | - |

##### 设置边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ListSetBorderColor(PX\_Object \*pListObject,px\_color color); |
| 功能 | 设置边框颜色 |
| Object | 对象指针 |
| index | 要设置的边框颜色 |
| 返回值 | - |

#### 虚拟键盘

##### 创建全键虚拟键盘

|  |  |
| --- | --- |
| 函数名 | PX\_Object\* PX\_Object\_VirtualKeyBoardCreate(px\_memorypool \*mp, PX\_Object \*Parent,px\_int x,px\_int y,px\_int width,px\_int height); |
| 功能 | 创建虚拟键盘 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 设置全键虚拟键盘背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_VirtualKeyBoardSetBackgroundColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置虚拟键盘背景颜色 |
| Object | 对象指针 |
| color | 背景颜色 |
| 返回值 | - |

##### 设置全键虚拟键盘边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_VirtualKeyBoardSetBorderColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置虚拟键盘边框颜色 |
| Object | 对象指针 |
| color | 边框颜色 |
| 返回值 | - |

##### 设置全键虚拟键盘指针颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_VirtualKeyBoardCursorColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置虚拟键盘指针颜色 |
| Object | 对象指针 |
| color | 指针颜色 |
| 返回值 | - |

##### 设置全键虚拟键盘按下颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_VirtualKeyBoardPushColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置虚拟键盘按下颜色 |
| Object | 对象指针 |
| color | 按下颜色 |
| 返回值 | - |

##### 键盘返回值

|  |  |
| --- | --- |
| 函数名 | px\_char PX\_Object\_VirtualKeyBoardGetCode(PX\_Object \*pObject); |
| 功能 | 取得键盘返回值 |
| Object | 对象指针 |
| 返回值 | 如果虚拟键盘有被按键,那么将会返回一个非0的ANSI字符(或控制符) |

##### 创建数字虚拟键盘

|  |  |
| --- | --- |
| 函数名 | PX\_Object\* PX\_Object\_VirtualNumberKeyBoardCreate(px\_memorypool \*mp, PX\_Object \*Parent,px\_int x,px\_int y,px\_int width,px\_int height); |
| 功能 | 创建数字虚拟键盘 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 设置数字虚拟键盘背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_VirtualNumberKeyBoardSetBackgroundColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置虚拟键盘背景颜色 |
| Object | 对象指针 |
| color | 颜色 |
| 返回值 | - |

##### 设置数字虚拟键盘边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_VirtualNumberKeyBoardSetBorderColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置虚拟键盘边框颜色 |
| Object | 对象指针 |
| color | 颜色 |
| 返回值 | - |

##### 设置数字虚拟键盘指针颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_VirtualNumberKeyBoardCursorColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置虚拟键盘指针颜色 |
| Object | 对象指针 |
| color | 颜色 |
| 返回值 | - |

##### 设置数字虚拟键盘按下颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_VirtualNumberKeyBoardPushColor( PX\_Object \*pObject,px\_color Color ); |
| 功能 | 设置虚拟键盘按下颜色 |
| Object | 对象指针 |
| color | 颜色 |
| 返回值 | - |

##### 键盘返回值

|  |  |
| --- | --- |
| 函数名 | px\_char PX\_Object\_VirtualNumberKeyBoardGetCode(PX\_Object \*pObject); |
| 功能 | 取得键盘返回值 |
| Object | 对象指针 |
| 返回值 | 如果虚拟键盘有被按键,那么将会返回一个非0的ANSI字符(或控制符) |

#### 勾选框

##### 创建勾选框

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_CheckBoxCreate(px\_memorypool \*mp, PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height,const char text[],PX\_FontModule \*fm); |
| 功能 | 创建勾选框 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| text | 勾选框文本 |
| fm | 字模 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_CheckBox \*PX\_Object\_GetCheckBox(PX\_Object \*Object); |
| 功能 | 取得勾选框对象数据 |
| Object | 对象指针 |
| 返回值 | 如果成功返回对象数据指针,否者返回PX\_NULL |

##### 取得勾选框勾选状态

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_Object\_CheckBoxGetCheck(PX\_Object \*Object); |
| 功能 | 取得勾选框勾选状态 |
| Object | 对象指针 |
| 返回值 | 如果被勾选状态返回PX\_TRUE,否者返回PX\_FALSE |

##### 设置背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_CheckBoxSetBackgroundColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置背景颜色 |
| Object | 对象指针 |
| clr | 背景颜色 |
| 返回值 |  |

##### 设置边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_CheckBoxSetBorderColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置背景颜色 |
| Object | 对象指针 |
| clr | 背景颜色 |
| 返回值 |  |

##### 设置按下颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_CheckBoxSetPushColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置按下颜色 |
| Object | 对象指针 |
| clr | 按下颜色 |
| 返回值 |  |

##### 设置指针颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_CheckBoxSetCursorColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置指针颜色 |
| Object | 对象指针 |
| clr | 指针颜色 |
| 返回值 |  |

##### 设置文本

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_CheckBoxSetText(PX\_Object \*Object,const px\_char text[]); |
| 功能 | 设置文本 |
| Object | 对象指针 |
| text | 文本 |
| 返回值 |  |

##### 设置文本颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_CheckBoxSetTextColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置文本颜色 |
| Object | 对象指针 |
| clr | 文本颜色 |
| 返回值 |  |

##### 设置勾选状态

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_CheckBoxSetCheck(PX\_Object \*Object,px\_bool check); |
| 功能 | 设置勾选状态 |
| Object | 对象指针 |
| check | PX\_TRUE表示勾选,PX\_FALSE表示非勾选 |
| 返回值 |  |

#### 菜单

##### 创建菜单

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_MenuCreate(px\_memorypool \*mp,PX\_Object \*Parent,px\_int x,int y,px\_int width,PX\_FontModule \*fontmodule); |
| 功能 | 创建菜单 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| fm | 字模 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得菜单根节点

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_Menu\_Item \* PX\_Object\_MenuGetRootItem(PX\_Object \*pMenuObject); |
| 功能 | 取得菜单根节点 |
| pMenuObject | 对象指针 |
| 返回值 | 菜单的根节点指针 |

##### 添加菜单节点

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_Menu\_Item \* PX\_Object\_MenuAddItem(PX\_Object \*pMenuObject,PX\_Object\_Menu\_Item \*parent,const px\_char Text[],PX\_MenuExecuteFunc \_callback,px\_void \*ptr); |
| 功能 | 添加菜单节点 |
| pMenuObject | 对象指针 |
| parent | 父节点指针 |
| Text | 节点文本 |
| \_callback | 回调函数,当菜单这个节点被点击时会执行该回调函数 |
| ptr | 回调函数指针 |
| 返回值 | 新创建的节点指针 |

#### 下拉框

##### 创建下拉选择框

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_SelectBarCreate(px\_memorypool \*mp,PX\_Object \*Parent,px\_int x,int y,px\_int width,px\_int height,PX\_FontModule \*fontmodule); |
| 功能 | 创建下拉框 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| fm | 字模 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 取得对象数据

|  |  |
| --- | --- |
| 函数名 | PX\_Object\_SelectBar \*PX\_Object\_GetSelectBar(PX\_Object \*pSelectBar); |
| 功能 | 取得对象数据 |
| pSelecrBar | 对象指针 |
| 返回值 | 如果成功返回对象数据指针,否者返回PX\_NULL |

##### 添加条目

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_Object\_SelectBarAddItem(PX\_Object \*PX\_Object\_SelectBar,const px\_char Text[]); |
| 功能 | 添加一个条目 |
| pSelecrBar | 对象指针 |
| Text | 条目文本 |
| 返回值 | 如果成功返回条目索引,否者返回-1 |

##### 移除条目

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SelectBarRemoveItem(PX\_Object \*PX\_Object\_SelectBar,px\_int index); |
| 功能 | 移除一个条目 |
| pSelecrBar | 对象指针 |
| index | 条目索引 |
| 返回值 | - |

##### 取得条目索引

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_Object\_SelectBarGetItemIndexByText(PX\_Object \*pObject,const px\_char Text[]); |
| 功能 | 通过条目文本搜索一个条目索引 |
| pSelecrBar | 对象指针 |
| index | 条目索引 |
| 返回值 | 如果找到返回条目索引,否者返回-1,如果有多个同名条目仅会返回最先的索引 |

##### 设置最大显示条目格式

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SelectBarSetDisplayCount(PX\_Object \*pObject,px\_int count); |
| 功能 | 设置最大显示条目格式,超过条目个数下拉框将会以滚动条的形式进行调整 |
| pSelecrBar | 对象指针 |
| count | 最大显示条目个数 |
| 返回值 | - |

##### 设置背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SelectBarSetBackgroundColor(PX\_Object \*pObject,px\_color color); |
| 功能 | 设置背景颜色 |
| Object | 对象指针 |
| clr | 背景颜色 |
| 返回值 |  |

##### 设置边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SelectBarSetBorderColor(PX\_Object \*pObject,px\_color color); |
| 功能 | 设置背景颜色 |
| Object | 对象指针 |
| clr | 背景颜色 |
| 返回值 |  |

##### 设置文本颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SelectBarSetFontColor(PX\_Object \*pObject,px\_color color); |
| 功能 | 设置文本颜色 |
| Object | 对象指针 |
| clr | 文本颜色 |
| 返回值 |  |

##### 设置指针颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_SelectBarSetCursorColor(PX\_Object \*pObject,px\_color color); |
| 功能 | 设置指针颜色 |
| Object | 对象指针 |
| clr | 指针颜色 |
| 返回值 |  |

#### 单选框

##### 创建单选框

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_RadioBoxCreate(px\_memorypool \*mp, PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height,const char text[],PX\_FontModule \*fm); |
| 功能 | 创建单选框 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| text | 单选框文本 |
| fm | 字模 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 单选框选中状态

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_Object\_RadioBoxGetCheck(PX\_Object \*Object); |
| 功能 | 取得单选框的选中状态 |
| Object | 对象指针 |
|  | 如果选中返回PX\_TRUE,否者PX\_FALSE |

##### 设置单选框背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_RadioBoxSetBackgroundColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置单选框背景颜色 |
| Object | 对象指针 |
| clr | 颜色 |
|  |  |

##### 设置单选框边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_RadioBoxSetBorderColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置单选框边框颜色 |
| Object | 对象指针 |
| clr | 颜色 |
|  |  |

##### 设置单选框按下颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_RadioBoxSetPushColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置单选框按下颜色 |
| Object | 对象指针 |
| clr | 颜色 |
|  |  |

##### 设置单选框指针颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_RadioBoxSetCursorColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置单选框指针颜色 |
| Object | 对象指针 |
| clr | 颜色 |
|  |  |

##### 设置单选框文本

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_RadioBoxSetText(PX\_Object \*Object,const px\_char text[]); |
| 功能 | 设置单选框文本 |
| Object | 对象指针 |
| text | 文本 |
|  |  |

##### 设置单选框文本颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_RadioBoxSetTextColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置单选框文本颜色 |
| Object | 对象指针 |
| clr | 文本颜色 |
|  |  |

##### 设置单选框选中状态

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_RadioBoxSetCheck(PX\_Object \*Object,px\_bool check); |
| 功能 | 设置单选框选中状态 |
| Object | 对象指针 |
| check | 选中状态 |
|  |  |

#### 文件浏览器

##### 创建文件浏览器

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_ExplorerCreate(px\_memorypool \*mp, PX\_Object \*Parent,px\_int x,px\_int y,px\_int Width,px\_int Height,PX\_FontModule \*fm,  PX\_ExplorerGetPathFolderCount \_func\_gpfdc,  PX\_ExplorerGetPathFileCount \_func\_gpfec,  PX\_ExplorerGetPathFolderName \_func\_gpfdn,  PX\_ExplorerGetPathFileName \_func\_gpfcn,  const px\_char path[260]  ); |
| 功能 | 创建文件浏览器 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| fm | 字模 |
| \_func\_gpfdc | 读路径文件夹个数的回调函数 |
| \_func\_gpfec | 读路径文件夹文件个数的回调函数 |
| \_func\_gpfdn | 读路径文件夹名称的回调函数 |
| \_func\_gpfcn | 读路径文件夹文件名的回调函数 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 设置文件浏览器边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerSetBorderColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置文件浏览器边框颜色 |
| Object | 对象指针 |
| clr | 颜色 |
|  |  |

##### 设置文件浏览器按下颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerSetPushColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置文件浏览器按下颜色 |
| Object | 对象指针 |
| clr | 颜色 |
|  |  |

##### 设置文件浏览器指针颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerSetCursorColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置文件浏览器指针颜色 |
| Object | 对象指针 |
| clr | 颜色 |
|  |  |

##### 设置文件浏览器文本颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerSetTextColor(PX\_Object \*Object,px\_color clr); |
| 功能 | 设置文件浏览器指针颜色 |
| Object | 对象指针 |
| clr | 颜色 |
|  |  |

##### 刷新文件浏览器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerRefresh(PX\_Object \*Object); |
| 功能 | 刷新文件浏览器 |
| Object | 对象指针 |
|  |  |

##### 文件浏览器选中个数

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_Object\_ExplorerGetSelectedCount(PX\_Object \*Object); |
| 功能 | 取得文件浏览器选中个数 |
| Object | 对象指针 |
| 返回值 | 选中个数 |

##### 文件浏览器路径

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerGetPath(PX\_Object \*Object,px\_char path[PX\_EXPLORER\_MAX\_PATH\_LEN],px\_int index); |
| 功能 | 取得文件浏览器选中路径 |
| Object | 对象指针 |
| index | 如果文件浏览器选中多个文件,则编码为0-N-1 |
| path | 输出路径 |
| 返回值 |  |

##### 打开文件浏览器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerOpen(PX\_Object \*Object); |
| 功能 | 打开文件浏览器 |
| Object | 对象指针 |
| 返回值 |  |

##### 关闭文件浏览器

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerClose(PX\_Object \*Object); |
| 功能 | 关闭文件浏览器 |
| Object | 对象指针 |
| 返回值 |  |

##### 设置文件浏览器的过滤参数

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerSetFilter(PX\_Object \*Object,const px\_char \*filter); |
| 功能 | 设置文件浏览器的过滤参数 |
| Object | 对象指针 |
| filter | 过滤参数,例如”\*.png\0\*.jpg\0”以\0隔开 |
| 返回值 |  |

##### 设置文件浏览器的最大选中大小

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_ExplorerSetMaxSelectCount(PX\_Object \*Object,int selectCount); |
| 功能 | 设置文件浏览器的最大选中大小 |
| Object | 对象指针 |
| selectCount | 最大选中大小,默认为1,多选则为多值 |
| 返回值 |  |

#### 子窗口

##### 创建子窗口

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_WidgetCreate(px\_memorypool \*mp,PX\_Object \*Parent,int x,int y,int width,int height,const px\_char title[],PX\_FontModule \*fontmodule); |
| 功能 | 创建子窗口 |
| mp | 内存池 |
| Parent | 父对象 |
| x,y | 平面坐标 |
| width,height | 宽度,高度 |
| title | 标题 |
| fm | 字模 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 显示子窗口

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetShow(PX\_Object \*pObject); |
| 功能 | 显示子窗口 |
| pObject | 对象指针 |
|  |  |

##### 隐藏子窗口

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetHide(PX\_Object \*pObject); |
| 功能 | 隐藏子窗口 |
| pObject | 对象指针 |
|  |  |

##### 隐藏/显示子窗口关闭按钮

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetShowHideCloseButton(PX\_Object \*pObject,px\_bool show); |
| 功能 | 隐藏/显示子窗口关闭按钮 |
| pObject | 对象指针 |
| show | 是否显示 |
|  |  |

##### 模态子窗口

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetSetModel(PX\_Object \*Object,px\_bool model); |
| 功能 | 设置是否为模态子窗口 |
| pObject | 对象指针 |
| show | 是否模态窗口 |
|  |  |

##### 取得子窗口的对象指针

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_WidgetGetRoot(PX\_Object \*pObject); |
| 功能 | 取得子窗口的对象指针 |
| pObject | 对象指针 |
|  |  |

##### 设置子窗体边框颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetSetBorderColor(PX\_Object \*pObject,px\_color clr); |
| 功能 | 设置子窗体边框颜色 |
| pObject | 对象指针 |
| clr | 要设置的颜色 |
|  |  |

##### 设置子窗体拖动条颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetSetBarColor(PX\_Object \*pObject,px\_color clr); |
| 功能 | 设置子窗体拖动条颜色 |
| pObject | 对象指针 |
| clr | 要设置的颜色 |
|  |  |

##### 设置子窗体背景颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetSetBackgroundColor(PX\_Object \*pObject,px\_color clr); |
| 功能 | 设置子窗体背景颜色 |
| pObject | 对象指针 |
| clr | 要设置的颜色 |
|  |  |

##### 设置子窗体焦点颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetSetFocusColor(PX\_Object \*pObject,px\_color clr); |
| 功能 | 设置子窗体焦点颜色 |
| pObject | 对象指针 |
| clr | 要设置的颜色 |
|  |  |

##### 设置子窗体标题文本颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetSetFontColor(PX\_Object \*pObject,px\_color clr); |
| 功能 | 设置子窗体标题文本颜色 |
| pObject | 对象指针 |
| clr | 要设置的颜色 |
|  |  |

##### 设置子窗体标题文本

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_WidgetSetTitle(PX\_Object \*pObject,const px\_char title[]) |
| 功能 | 设置子窗体标题文本 |
| pObject | 对象指针 |
| title | 窗体文本 |
|  |  |

#### 消息对话框

##### 创建消息对话框

|  |  |
| --- | --- |
| 函数名 | PX\_Object \* PX\_Object\_MessageBoxCreate(px\_memorypool \*mp,PX\_Object \*parent,PX\_FontModule \*fontmodule); |
| 功能 | 创建消息对话框 |
| mp | 内存池 |
| Parent | 父对象 |
| fontmodule | 字模 |
|  | 如果创建成功返回对象指针,否者返回PX\_NULL |

##### 关闭消息对话框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_MessageBoxClose(PX\_Object \*pObject); |
| 功能 | 关闭消息对话框 |
| pObject | 对话框对象 |
|  |  |

##### 确定消息对话框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_MessageBoxAlertOk(PX\_Object \*pObject,const px\_char \*message,PX\_Object\_MessageBoxCallBack func\_callback,px\_void \*ptr); |
| 功能 | 弹出一个确定消息对话框 |
| pObject | 消息对话框对象 |
| message | 显示文本 |
| func\_callback | 点击确定时回调函数 |
| ptr | 用户指针 |

##### 纯消息对话框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_MessageBoxAlert(PX\_Object \*pObject,const px\_char \*message); |
| 功能 | 弹出一个仅显示文本消息对话框,这个对话框需要手动关闭 |
| pObject | 消息对话框对象 |
| message | 显示文本 |

##### 确定/取消消息对话框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_MessageBoxAlertYesNo(PX\_Object \*pObject,const char \*Message,PX\_Object\_MessageBoxCallBack func\_yescallback,px\_void \*yesptr,PX\_Object\_MessageBoxCallBack func\_nocallback,px\_void \*noptr); |
| 功能 | 弹出一个确定/取消消息对话框 |
| pObject | 消息对话框对象 |
| message | 显示文本 |
| func\_yescallback | 点击确定时的回调函数 |
| yesptr | 确定的用户指针 |
| func\_nocallback | 点击取消的回调函数 |
| noptr | 取消的用户指针 |

##### 输入消息对话框

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_Object\_MessageBoxInputBox(PX\_Object \*pObject,const char \*Message,PX\_Object\_MessageBoxCallBack func\_yescallback,px\_void \*yesptr,PX\_Object\_MessageBoxCallBack func\_cancelcallback,px\_void \*cancelptr); |
| 功能 | 弹出一个输入消息对话框,允许用户输入一行的文本数据 |
| pObject | 消息对话框对象 |
| message | 显示文本 |
| func\_yescallback | 点击确定时的回调函数 |
| yesptr | 确定的用户指针 |
| func\_nocallback | 点击取消的回调函数 |
| noptr | 取消的用户指针 |

##### 取得输入消息对话框的文本数据

|  |  |
| --- | --- |
| 函数名 | px\_char \* PX\_Object\_MessageBoxGetInput(PX\_Object \*pObject); |
| 功能 | 取得输入消息对话框的文本数据 |
| pObject | 消息对话框对象 |
| message | 显示文本 |
| 返回值 | 文本指针 |

### PainterEngine UI设计框架

#### 基本数据类型

PainterEngine 可以通过定义PX\_UI从JSON文件中加载一个UI布局,其JSON的格式规范如下

“控件类型” //必选参数,控件类型

{

“id”:”控件id”, //可选参数,控件id

“x”:”控件的x坐标”, //可选参数,控件x坐标,默认为0

“y”:”控件的y坐标”,// 可选参数,控件y坐标,默认为0

“width”:”控件宽度”,//可选参数,控件宽度,默认为0

“height”:”控件高度”,//可选参数,控件高度,默认为0

“length”:”控件长度”,//可选参数,控件长度,默认为0

“halign”:”水平对齐方式”,//可选参数,水平对齐方式

“valign”:”水平垂直方式”,//可选参数,水平垂直方式

}

其中,UI中的基本参数为所有的UI控件都有的参数类型,包括以下

|  |  |  |  |
| --- | --- | --- | --- |
| UI参数名 | 类型 | |  |
| id | string | | 控件id,必须唯一 |
| x | number | | 可选参数,控件x坐标,默认为0 |
| y | number | | 可选参数,控件y坐标,默认为0 |
| width | number | | 可选参数,控件宽度,默认为0 |
| height | | number | 可选参数,控件高度,默认为0 |
| length | | number | 可选参数,控件长度,默认为0 |
| halign | | string | 可选参数,水平对齐方式left/mid/right |
| valign | | string | 可选参数,水平垂直方式left/mid/right |

#### UI类型及对应属性

目前,PainterEngine UI设计框架支持以下几种控件类型,同时其支持属性如下

#### label静态文本框

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| bordercolor | number数组 | 边框颜色,格式为[a,r,g,b] |
| style | string | 风格,如果为round则为圆角类型 |
| border | bool | 是否有边框 |
| align | string | 文本对齐方式  lefttop 左上角对齐  leftmid 左居中对齐  leftbottom 左下对齐  midtop 居中置顶对齐  center 居中对齐  midbottom 居中置底对齐  righttop 右上角对齐  rightmid 靠右居中对齐  rightbottom 右下角对齐 |

#### processbar进度条

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| bordercolor | number数组 | 边框颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| border | bool | 是否有边框 |
| max | number | 最大值 |

#### image图像框

|  |  |  |
| --- | --- | --- |
| 扩展属性 |  |  |
| 无 |  |  |

#### sliderbar 滑动框

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| max | number | 最大值 |
| min | number | 最小值 |
| type | string | vertical 垂直滑动框  其它 水平滑动框 |
| style | string | liner 线性滑动框类型  其它 方框滑动框 |
| color | number数组 | 颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| buttonlength | number | 滑动块尺寸 |

#### pushbutton/cursorbutton按钮

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| fontcolor | number数组 | 文本颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| pushcolor | number数组 | 按下颜色,格式为[a,r,g,b] |
| bordercolor | number数组 | 边框颜色,格式为[a,r,g,b] |
| style | string | round圆角按钮  其它 矩形按钮 |
| border | bool | 是否具有边框 |
| text | string | 按钮文本 |

#### edit文本编辑框

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| fontcolor | number数组 | 文本颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| cursorcolor | number数组 | 指针颜色,格式为[a,r,g,b] |
| border | bool | 是否具有边框 |
| passwordstyle | bool | 是否为密码样式,如果是,输入的文本都显示为\*号 |
| autonewline | bool | 是否自动换行 |
| limit | string | 限制文本,仅允许输入限制文本中的字符 |
| maxlength | number | 最大长度 |
| style | string | round圆角编辑框  其它 矩形编辑框 |

#### scrollarea滚动区域

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| bordercolor | number数组 | 边框颜色,格式为[a,r,g,b] |
| border | bool | 是否具有边框 |

#### autotext自动换行静态文本框

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| text | string | 文本 |
| fontcolor | number数组 | 文本颜色 |

#### virtualkeyboard虚拟键盘

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| fontcolor | number数组 | 文本颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| cursorcolor | number数组 | 指针颜色,格式为[a,r,g,b] |
| pushcolor | number数组 | 按下颜色,格式为[a,r,g,b] |
| border | bool | 是否具有边框 |

#### virtualnumberkeyboard虚拟数字键盘

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| fontcolor | number数组 | 文本颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| cursorcolor | number数组 | 指针颜色,格式为[a,r,g,b] |
| pushcolor | number数组 | 按下颜色,格式为[a,r,g,b] |
| border | bool | 是否具有边框 |

#### checkbox 选择框

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| fontcolor | number数组 | 文本颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| cursorcolor | number数组 | 指针颜色,格式为[a,r,g,b] |
| pushcolor | number数组 | 按下颜色,格式为[a,r,g,b] |
| bordercolor | number数组 | 边框颜色,格式为[a,r,g,b] |
| border | bool | 是否具有边框 |
| text | string | 选择框文本 |

#### radiobox单选框

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| fontcolor | number数组 | 文本颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| cursorcolor | number数组 | 指针颜色,格式为[a,r,g,b] |
| pushcolor | number数组 | 按下颜色,格式为[a,r,g,b] |
| bordercolor | number数组 | 边框颜色,格式为[a,r,g,b] |

#### selectbar下拉框

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| fontcolor | number数组 | 文本颜色,格式为[a,r,g,b] |
| backgroundcolor | number数组 | 背景颜色,格式为[a,r,g,b] |
| cursorcolor | number数组 | 指针颜色,格式为[a,r,g,b] |
| bordercolor | number数组 | 边框颜色,格式为[a,r,g,b] |
| style | string | round圆角下拉框  其它 矩形下拉框 |
| displaycount | number | 下拉最大显示个数,超出个数将使用滚动滑块滚动显示 |
| items | string数组 | 条目文本,以0开始编号 |
| currentindex | number | 当前条目,如没有默认为0 |

#### Widget 子窗体

|  |  |  |
| --- | --- | --- |
| 扩展属性 | 类型 |  |
| cursorcolor | number数组 | 指针颜色,格式为[a,r,g,b] |
| pushcolor | number数组 | 按下颜色,格式为[a,r,g,b] |
| bordercolor | number数组 | 边框颜色,格式为[a,r,g,b] |

### 游戏世界框架

#### 游戏世界初始化

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_WorldInitialize(px\_memorypool \*mp,PX\_World \*World,px\_int world\_width,px\_int world\_height,px\_int surface\_width,px\_int surface\_height,px\_dword calcsize); |
| 功能 | 初始化一个游戏世界 |
| mp | 游戏世界内存池 |
| World | 世界实例 |
| world\_width | 世界宽度(注意,超出世界宽度的Object将不再参与物理计算) |
| world\_height | 世界高度(注意,超出世界高度的Object将不再参与物理计算) |
| surface\_width | 视窗宽度大小 |
| surface\_height | 视窗高度 |
| calcsize | 用于物理计算的内存大小(树优化,例如碰撞检测) |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 取得游戏世界Object的个数

|  |  |
| --- | --- |
| 函数名 | px\_int PX\_WorldGetCount(PX\_World \*World); |
| 功能 | 取得游戏世界Object的个数 |
| World | 世界实例 |
| 返回值 | Object的个数 |

#### 删除Object

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldRemoveObject(PX\_World \*world,PX\_Object \*pObject); |
| 功能 | 从游戏世界删除一个Object |
| World | 世界实例 |
| pObject | 要删除的指针 |

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldRemoveObjectByIndex(PX\_World \*world,px\_int i\_index); |
| 功能 | 从游戏世界删除一个Object |
| World | 世界实例 |
| i\_index | Object的索引 |

#### 在世界查找Object

|  |  |
| --- | --- |
| 函数名 | //LIMIT-Only used to ObjectUpdate Function  \_LIMIT px\_int PX\_WorldSearchRegion(PX\_World \*world,px\_float centerX,px\_float centerY,px\_float raduis,PX\_Object \*Object[],px\_int MaxSearchCount,px\_dword impact\_test\_type); |
| 功能 | 划出一个圆形区域,查找区域内的所有Object |
| World | 世界实例 |
| centerX | 区域圆点X坐标 |
| centerY | 区域圆点Y坐标 |
| Object | 返回数组指针 |
| MaxSearchCount | 最大搜索个数 |
| impact\_test\_type | 碰撞过滤器类型 |
| 注意 | 这个函数仅允许在PX\_ObjectUpdate中被调用 |

#### 添加Object

|  |  |
| --- | --- |
| 函数名 | px\_bool PX\_WorldAddObject(PX\_World \*World,PX\_Object \*pObject); |
| 功能 | 在游戏世界中添加一个Object |
| World | 世界实例 |
| pObject | 要添加的Object指针,添加到游戏世界以后,该Object内存由游戏世界接管 |
| 返回值 | 如果成功返回PX\_TRUE,否者PX\_FALSE |

#### 更新游戏世界

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldUpdate(PX\_World \*World,px\_uint elpased ); |
| 功能 | 更新游戏世界 |
| World | 世界实例 |
| elapsed | 上一次update经过的毫秒时间 |

#### 渲染游戏世界

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldRender(px\_surface \*psurface,PX\_World \*World,px\_uint elpased ); |
| 功能 | 渲染游戏世界 |
| World | 世界实例 |
| elapsed | 上一次render经过的毫秒时间 |

#### 设置摄像机

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldSetCamera(PX\_World \*World,px\_point camera\_center\_point); |
| 功能 | 设置摄像机的中心位置 |
| World | 世界实例 |
| camera\_center\_point | 摄像机指向的中心位置 |

#### 设置游戏对象的碰撞类型

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldSetImpact(PX\_Object \*pObj,px\_dword type,px\_dword impact); |
| 功能 | 设置游戏对象的碰撞类型 |
| pObj | 需要设置的游戏对象 |
| type | 对象类型,该对象与type&impact!=0的对象发生碰撞 |
| impact | 碰撞类型, 该对象与type&impact!=0的对象发生碰撞 |
| 备注 | 当一个对象与另一个对象发生碰撞后,两个对象都会收到一个PX\_OBJECT\_EVENT\_IMPACT事件,可以通过PX\_Object\_Event\_GetPtr来获取另一个与自己发生碰撞的对象指针 |

#### 设置游戏世界辅助线间隔

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldSetAuxiliaryXYSpacer(PX\_World \*pw,px\_int x,px\_int y); |
| 功能 | 设置游戏世界辅助线 |
| pw | 游戏世界实例 |
| x | 水平间隔 |
| y | 垂直间隔 |

#### 设置游戏世界辅助线

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldEnableAuxiliaryLine(PX\_World \*pw,px\_bool bline); |
| 功能 | 设置游戏世界辅助线 |
| pw | 游戏世界实例 |
| bline | 是否显示世界辅助线,PX\_FALSE为不显示,否则显示 |

#### 设置游戏世界辅助线的颜色

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldSetAuxiliaryLineColor(PX\_World \*pw,px\_color color); |
| 功能 | 设置游戏世界辅助线 |
| pw | 游戏世界实例 |
| color | 辅助线颜色 |

#### 世界坐标转换为屏幕坐标

|  |  |
| --- | --- |
| 函数名 | px\_point PX\_WolrdObjectXYtoScreenXY(PX\_World \*pw,px\_float x,px\_float y); |
| 功能 | 世界坐标转换为屏幕坐标 |
| x,y | 游戏世界左边 |
| 返回值 | 屏幕坐标 |

#### 释放游戏世界

|  |  |
| --- | --- |
| 函数名 | px\_void PX\_WorldFree(PX\_World \*pw); |
| 功能 | 释放游戏世界 |
| pw | 游戏世界实例 |