1. 设计篇 （使用设计模式ui拖拽，使用exml代替显示对象，使用id引用对象）
2. 代码篇
3. 项目结构与文件 源码src 资源 项目配置文件（包括第三放库） 资源引入文件
4. 配置说明
5. 命令行 （2）入口文件
6. 显示对象和容器 所有显示对象父类 容器 包括Shape Sprite(属于容器)

属性:x,y,scalex,scaley,alpha,rotation,visible width height anchorOffsetX

包括位图 文本 形状 舞台this.stage

自定义显示对象（继承，然后添加属性如血条，蓝条，或者作为容器，其中渲染多种对象）

必须加入到显示列表才能显示出来this.addChild 最开始的类是指舞台this是舞台

锚点：指的是显示对象的”重心”，用于缩放和旋转，默认是设在左上，很多时候要改到中间（圆shape默认是以圆心为中心）

平移x 本地坐标：相对于父显示对象 舞台坐标 LocalToGlabol

允许点击 .touchEnable 斜切skewX

容器中加入显示对象容器.addChild removeChild

子对象数numChildred 容器深度（包括舞台也是个容器） 子对象第一个为0，第2个为1

指定深度 addChildAt removeChildAt(1)只用数值 这里的深度值也可当作id了（容器范围内） 移除所有子对象 removeChildren 交换对象 swapChildren(a,b) 按深度交换 swapChildrenAt() 设置深度setChildIndex 获取子对象 getChildAt() getChildByName 根据name属性

1. 绘图 绘图对象 .graphics 绘图方法 开始填充(初始化) beginFill() 结束填充endFill() 设置描边lineStyle(width,color) 画圆drawCircle(rx,ry,radius)

画线 设置线条样式 moveTo 移动锚点 lineTo 绘制的时候锚点自动移动到终点，可以连续lineTo画折线

曲线 二次贝赛尔曲线 moveTo起点 curveTo(x1,y1,x2,y2) 控制点 终点

圆弧drawArc(x,y,radius,startAngle,endAngle,direct) 圆心 半径 起角 终角 可填充画半圆等 也可不填充画空心

拱形

扇形 进度条 不规则边框进度条，用遮罩

绘制多个形状 可以多次endfill

注意图像不会自动消失，可以clear 清屏

1. 遮罩

矩形遮罩 shp.mask=new egret.Rectangle(x,y,w,h ) 增加遮罩后只有遮罩部分可显示，其他隐藏

对象遮罩，一个对象作为另一个的遮罩 sp1.mask=sp2 只有遮罩对象的不透明区可显示，其他隐藏 删除遮罩 mask=null

矩形遮罩不能动态修改，只能新建重赋值 对象遮罩不加入显示列表的话不能修改，加入显示列表后可以动态修改，实现交互遮罩

1. 碰撞检测

点碰撞 shp.hitTestPoint（包围盒） 像素级点碰撞 hitTestPoint(x,y,true)

矩形碰撞检测

1. 文本Textfiled 文本.text 输入文本 type=egret.TextFieldType.INPUT

坐标 宽高 文本颜色textColor 获取焦点.setFocus() 输入样式 .inputType=文本egret.TextFieldInputType.TEXT 密码 .PASSWORD .displayAsPassword=true

电话.TEL

位图文本 BitmapText 加载字体文件getRES .font=

字体：fontFamily=”” 自定义字体 registerFontMapping(name,path) 需要先加载

字号 .size 自身尺寸可以设置，默认是按照首次文本内容大小

描边 宽度stroke 描边颜色 strokeColor

文本对齐textAlign egret.HorizontalAlign.CENTER; 右对齐.RIGHT

垂直对齐verticalAlign egret.VerticalAlign.MIDDLE; 居中 底.BUTTOM

加粗bold 斜体italic

多种样式混合 使用属性数组 .textFlow=[{text,style:{size}}]

超链接 textFlow中某段文字设置href {href:event:} 监听addEventListener( egret.TextEvent.LINK, function( evt:egret.TextEvent )

打开url：{href:url}

全局默认大小，默认颜色

1. 事件机制 发送者 侦听者 lisener 事件对象event 包含数据

设置事件 事件类继承egret.Event 继承父构造 加入自己的各种数据属性 然后由发送者 dispatchEvent(myevent) new事件，设置对象属性，发送这个事件对象

接收方监听某个类型的事件 ，addEvenLisener注册一个处理函数 当收到事件的时候才执行该函数， 并从收到的event对象中 获取信息

常见事件：加入舞台

构造函数 参数: 类型 冒泡方式 是否可取消 事件target属性

Target是指发送者？ 鼠标点击

addEventLisener() 可以设置优先级

侦听器（处理函数）必须返回空 是否注册了侦听器？hasEventLisener（type）

touchEnable

触摸事件TOUCH\_BEGIN TOUCH\_MOVE TOUCH\_TAP TOUCH\_END

取消触摸TOUCH\_CANCEL

网络

发送http new HttpRequest 对象 设置响应类型responeType open(url,method) setRequestHeader 设置请求头

Send(数据) 监听请求成功 response属性 ，错误 进度 event.bytesLoaded .bytesTotal egret.ProgressEvent IOErrorEvent egret.Event.COMPLETE,

加载位图

ImageLoader

.once( )complete事件 .load(url) 处理函数中 loader.data 或者event.currentTarget.data就是位图数据

new Bitmap(bitmapdata/texture) 或者用纹理 .texture 纹理对象 new Texture(),bitmapdata

加载文本和二进制 还是用HttpRequest(xhr) 默认加载类型就是text

responseType= egret.HttpResponseType.ARRAY\_BUFFER;

egret.HttpMethod.GET

1. 位图和纹理

纹理 bitmap.texture=RES.getRes(“id”) 在资源json中配置 name

九宫格 .scale9Grid=new egret.Rectangle() 用方块设置九宫

纹理填充方式(图片大小小于对象大小时) 默认是拉伸填充 重复填充fillMode= egret.BitmapFillMode.REPEAT

纹理集 json 分解成不同碎片 碎片名 x y w h 偏移（一般是0） 源宽高（一般和w h 相等） type:sheet 获取某块 .texture=RES.getRes(“资源id#碎片名”)

截屏 动态纹理new egret.RenderTexture().drawToTexture(sprite) 显示对象变纹理

Bitmap.texture=rendertexture

转换成base64 .toDataURL(image/png,rect) 类型 截取区域（矩形） 默认是整个texture 保存为文件saveToFile() 浏览器是可以读取本地图片格式的，也可下载任意图片，但是路径只能由浏览器定

截屏只用截下整个舞台

1. 混合模式

覆盖 默认就是覆盖 叠加shape.blendMode= egret.BlendMode.ADD

擦除 .ERASE

1. 滤镜

发光 new egret.GlowFilter（color,alpha,blurx,blury） .filters=[filter1]

颜色 透明度 模糊

颜色矩阵

自定义shader

1. 时间控制

Timer(delay,count) 间隔毫秒 次数 0为无限次 start reset stop

事件TimerEvent.TIMER TimerEvent.TIMER\_COMPLETE

心跳 （固定60桢）

Egret.startTick(func,this) stopTick 回调有返回 如果为true会重绘 否则不重绘

Egret.getTimer() 获取时间

桢事件ENTER\_FRAME

1. 多媒体

音频： new egret.Sound sound.load(url) complete事件

用urlloader加载 complete loader.load(new URLRequest(url)) loader.dataFormat

Sound=loader.data

Sound=getRes(“id”)

播放 play(start,loop) 0为无限次 返回一个声音控制对象SoundChannel 暂停重播 position channel监听完成事件

视频

环境信息egret.Capabilities.isMobile 是否手机 .language语言 .os系统 .runtimeType 项目运行类型web native runtime2 wxgame

陀螺仪new egret.DeviceOrientation().start() CHANGE事件 方向 e.alpha e.beta e.gamma 三轴旋转角度

Gps new egret.Geolocation().start Event.CHANGE 海拔e.altitude纬度 .latitude 经度.longtitude 速度.speed

错误处理，拒绝访问gps egret.GeolocationEvent.PERMISSION\_DENIED

失败egret.GeolocationEvent.UNAVAILABLE

渲染模式

Webgl渲染 egret.runEgret({renderMode:"webgl"});

egret.Capabilities.renderMode

1. 屏幕适配

缩放模式 在index.hrml中设置 或者在this.stage.scaleMode= egret.StageScaleMode.SHOW\_ALL indexl.html 设置舞台默认大小

然后会根据屏幕适配 showall 等比缩放 显示居中 noscale 不根据屏幕适配 noborder 裁掉两边 exactfit 非等比缩放，直接填充屏幕 fixedWidth

旋转模式 auto 从上到下 portrait 固定竖屏 landscape固定横屏

1. 调试

If(Debug){} 该代码块 发行时会被完全移除 内部测试代码只有debug模式可用

Fps data-show-fps="true/false"

Log 日志 egret.log() 可以传入任意个参数

1. 原生js
2. 库
3. RES加载 可以用promise 或者是await语法 把异步调用同步化

RES.loadConfig().then().then 注解语法@RES

资源生命周期

处理器 自定义

资源解析器 json 根据不同type 不同解析器RES.registerAnalyzer(type, analyzerClass)

内置解析器有二进制 文本 纹理 string json 纹理集 位图纹理 sound xml 可以在加载文件时自动转换

图片可以加入九宫格{“scale9grid”:””} 声音加入声音类型 soundType:music effect

如何读取解析二进制 type bin 得到一个字节数组ByteArray

new egret.ByteArray( RES.getRes("bg") ). .readUnsignedShort().toString(16)

字节数组转换成16进制整数

对比编码规范

缓存：加载成功后 以后再加载只会从缓存读取（不会再请求url） 清除

RES.destroyRes(“id”)

资源获取 getRes() 同步加载必须先loadGroup预加载 异步加载getResAsync

getResByUrl(url,func,this,method) 配置中不存在的直接url加载

预加载资源组 loadGroup GROUP\_COMPLETE GROUP\_PROGRES GROUP\_LOAD\_ERROR

setProgress(evt.itemsLoaded,evt.itemsTotal) 多次失败 计数

默认配置 loadConfig(url,baseurl) CONFIG\_COMPLETE CONFIG\_LOAD\_ERROR

1. EUI eui大部分是拖动出来的 是exml 和ts一套的 实际上exml也会被解析为ts

Eui和普通显示对象混合添加 自动布局

Exml格式 其实就是显示对象的嵌套 会自动添加到显示列表

每个exml被称为皮肤 和使用皮肤的组件（如button）分离 ，但是exml/skin本身可以看成是组件的子显示对象，包含文字，图片，布局等更下层的显示对象 ，这些文字，图片都是可以获取和修改的 而控制逻辑层一般都是写在最上层的父组件中

E:Skin e:Image e:Label 每层都可以给予class id等属性 方便在ts中引用

Source size width

Exml不一定就是皮肤，自然也可以是一个完整的组件，e:Button 不仅如此，甚至可以在组件下层继续加入组件 e:Slider

加载exml .skinName=”” 动态加载(非皮肤) EXML.load(url,回调) skinname=evt

直接嵌入exml代码 `` 用模板字符串 反引号 var exml=”” skinName=

Class属性是指全局类名 会被解析成该类（必须）

子项相当与 new eui.Image() this.addChild id属性设置后id 就是子项的变量名

数据绑定 text=”{data.label}” 视图状态 根据状态自动改变显示的子项或者属性 includeIn=”状态名”

皮肤分离： 使用组建拖拽创建时 一个exml可以可视编辑

一个ts: ts类 对象属性只要设成子部件id 就可以操控

createdChilred(){} 中写

如果自己创建了exml 在ts 中使用class类名.继承eui.Component consturctor(){super 并且加载this.skinName } 监听UIEvent.COMPLETE

CreateChildren(){super.createChildren() }

可以通过主题使用，很多组件有默认的皮肤， json中 组件类名+skinName值

自动布局 舞台尺寸 this.stage.stageHeight 匹配舞台和屏幕

组件和舞台的适应 eui中可以用百分比 所有属性都是相对父容器 eui.UILayer宽高自动和舞台一致

失效验证 改变属性后立刻重绘 还是等到改变一定次数再重绘 失效验证不重绘，只用变量标注属性已改变 父布局改变，子跟随

自动布局就是 事件发生时 自动设置子对象的x,y,width,height measure测量属性 updateDisplayList 设置属性

Group是容器，使用布局属性 layout 实现解耦合 layout.measure layout.updateDisplayList

异步：更新时的延迟错乱 需要推迟显示或者提前算

自动布局的调试属性

默认的几种状态

标记状态失效 都是默认继承的

用某事件触发this.invalidateState()调用getCurrentState() 失效 按下 弹上 通过自己重写可以扩展状态类型 if() return “状态”

button.currentState = "down"

布局：一般来说布局不需要写代码的，直接在设计界面就行了

基本布局(默认)：.layout=new eui.BasicLayout() 绝对定位 ，按照父对象的相对坐标 子x,y 居中.horizontalCenter=0 .verticalCenter=0 相对于中心的值 边距top left

水平布局：和垂直布局忽略 x,y gap间隔 horizontalAlign水平对齐（左中右），verticalAlign(顶底中) padding内边距

垂直同上

网格布局 new GridLayout() horizontalGap水平间隔 verticalGap 垂直间隔 conlumnAlign列对齐 rowAlign行对齐 padding

控件 （都有默认皮肤，可以修改设置为新皮肤） 建议用拖拽

标签lable 图片Image 按钮 默认可点击 获取按钮文本 button.labelDisplay.text 复选框 .selectd属性 UIEVENT.CHANGE事件

和togglebutton一样 单选框 附加数据 selected value 单选按钮组 直接设个同样的组名 .groupName=“a” 必须每个都加上监听change事件

单选组方法2：new eui.RadioButtonGroup() .group=组 只用给组加上监听

.selectedValue

滑块 水平垂直 Hslider Vslider maximum minimum value 最大 最小值 监听 UIEVENT.CHANGE

进度条 maximum minimum value 利用timer模拟？

输入文本 .text默认文本

容器：简单容器和sprite一样 group getElementAt numElements layout

内容超出容器 .scrollEnabled=true 就不显示了

层叠容器 多个子项（子对象）只显示一项 addChild .selectedIndex=1 (选中的项的深度 ) 通常配合选项卡 使用

面板 带标题和内容 关闭按钮 .title .skinName id titleDisplay closeButton moveArea

滚动控制容器 Scroller 要滚动的容器(group或者list)加入到viewport属性 控制滚动 .viewport.scrollv=10 停止滚动动画.stopAnimation

取消滚动条 .verticalScrollBar.autoVisibility自动隐藏 .visble隐藏

数据容器 需要绑定 数据源和子项样式

数据源不能用普通数组 要用数组集合

dataProvider=new eui.ArrayCollection(array)

子项样式模板ts 类继承eui.ItemRender 加入子显示对象 重写dataChanged方法：子对象属性=this.data.属性 这样就实现动态绑定了，数组变化

.itemRenderer=

数组集合

使用的数组是[{name: value:},{} ] 的格式 每个数组项都是对象，带有属性

eui.CollectionEvent.COLLECTION\_CHANGE 监听数据改变事件

增加数据addItem({}) addItemAt() 加到某个索引（可覆盖） getItemAt() .length .getItemIndex(a) 这里面的对象最好是个引用吧，{}新对象不合适

ReplaceItemAt() removeAll() removeItemAt()

列表 list 项被点中 eui.ItemTapEvent.ITEM\_TAP事件 选中项 selectedIndex .selectedItem 多选模式allowMultipleSelection = true

.selectedIndices Items 数组

选项卡 TabBar 数据源是viewStack 或者设置为数组集合

1. GAME库

桢动画movieclip 用swf gif转 或者制作 纹理集和json

动作：帧率 某帧事件 工厂类 加载png和json getRes

new egret.MovieClipDataFactory( json, png );

获取动作 new egret.MovieClip( mcFactory.generateMovieClipData( "run" ) );

播放：从桢标签 gotoAndPlay(“标签”，次数) 帧数播放 （3）

事件 侦标签.MovieClipEvent.FRAME\_LABEL e.type e.frameLabel

完成 一次完成.Event.LOOP\_COMPLETE 全部完成.Event.COMPLETE 事件。

滚动视图 new egret.ScrollerView() .setContent(显示对象) 默认和内容一致 .width .height 内容不需要加入到显示列表 回弹bounces=false (出界拉回) 滚动策略 水平 垂直 默认auto

事件egret.Event.CHANGE Event.COMPLETE

超时触发器setTimeout(fn,this,1000) clearTimeout(id)

URL类 请求 加载 加载数据格式化 请求方法

Loader=new egret.URLLoader newURLRequest(url) .load(request)

COMPLETE监听 loader.data

数据格式loader.dataFormat= egret.URLLoaderDataFormat.VARIABLES;

请求方法request.method= egret.URLRequestMethod.POST

发送数据request.data= new egret.URLVariables("test=ok")

1. 缓动TWEEN库

Tw=egret.Tween.get(shp).to({x:100},100) 属性值缓慢变化 get(shp,{loop:true})循环

缓动变化事件

过程参数to(Ease.backInOut)

回调.call(fn) 等待.wait()

1. Websocket库

New egret.Websocket Event.CONNECT ProgressEvent.SOCKET\_DATA 收到消息

连接服务器.connect(url,port) .writeUTF() 获取消息.readUTF()

1. P2物理引擎

刚体 形状 世界 世界步进.step 不动刚体进入睡眠模式 绑定显示对象

获取刚体数组 获取形状数组 显示对象和刚体同步绑定坐标，需要换算 显示对象的中心

1. 粒子

粒子系统 总时间 间隔 纹理 x,y 最大粒子数 构造 start() stop() changeTexture

重力粒子 x y 存活时间 角度 速度 水平重力 径向加速度 切向加速度 出现大小 出现时旋转 出现时alpha

加载纹理 配置 json.包含各种粒子属性 new System(texture,config)

system.start .stop 系统加到舞台addChild

1. JSZIP

读取zip getResAsync(zip).then((data)=>{JZIP.loadAsync(data) }).then( zipdata.file(文件名).async(“text”) 类型 ).then(text)

压缩

1. Tilemap 层 ： 每一个tile是一个显示对象？ Tmx需要解码xml和base64

地图属性：旋转 地图宽高（格子数） 格子宽 地图格式(xml,base64,csv) 渲染顺序 背景颜色

图层：图像 对象（一般是用来绘制陷阱或者可动物的） 块/地形

修改格式 <tileset> request请求 event.currentTarget.responese 用XML.parse解析

new tiled.TMXTilemap() .render() 可以点击 、监听

图块Tileset 层Layer 对象层ObjectGroup 图像ImageLayer 属性Property

背景colorLayer 地图和格子 Tilemap 中的Tile对象

五类对象：矩形 椭圆 多边 折线 PolyLine图片对象TMXImage

创建图像 获取层 获取格子 uid 坐标转数组

滚动地图 x 碰撞，网格为单位，对象层为单位

Tilemap 属性： 列数行数 baseURL 格子宽高tileheight tilewidth 获取所有图层 getlayers() getObjects()

层属性： 层名 行 列格子数 透明度 tile宽高 图层属性properties

根据坐标获取tile getTile(x,y) 获取tile id getTileId（x,y） 清除tile clearTile(x,y)

setTile(tilex,tiley,id)

格子属性 动画annimation gid格子id image 格子坐标（索引 ）tileX tileY

Tilemap地图 tileset所有的格子

对象 :gid set中会分配id id对象自己的id 是否椭圆 是否图像 是否折线 是否多边形 对象名name 对象类型type 层深 z 对象是显示对象

Gid是从tileset第一个

对象组 getObjectById() ByIndex removeObjectById

图像层 alpha bitmap texture

Base64解析

四、更多常用API 属性和事件

1、类 位图数据（像素点）bitmap(传入位图数据或纹理) bitmap.bitmapData

加载图片可以保存为纹理或者位图数据

纹理对象 texture.bitmapData=bmd 位图数据是个数组32位 rgba

通用的方法 显示方法 ，继承的方法 clone()克隆

XML对象

显示对象 .matrix .parent容器 getBounds() localtoglobal hasEventLisener

点

矩形Rectangle top right left width .contains() 是否包含点 containsPoint() containsRect() 是否包含矩形 copyFrom 增加大小 .intersects(另一个矩形 )

2、事件 事件类别 加载事件 完成 失败 点击事件 进度事件 eui事件 改变事件

Once() 只回调一次

remove remove\_from\_stage

类型字符串 数组的操作 转换

1. 一些技巧 （游戏设计方面和代码方面）

一个ts游戏程序的执行流程： 加载配置json 加载组

**尽量用函数封装语句，一个函数只做一件事，复杂功能应该转交给别的函数**

**比如飞行中使用碰撞检测 fly(){ hittest()} hittest(){} 把复杂的函数拆分成多个函数**

碰撞检测：是人碰撞东西还是东西碰撞人 ，明雷，暗雷

给静态地图对象或者怪物加上对人的检测 反向思维，可以解除一个对象过多的判断

ishitPlane() ishitGroup() ishitmonster()

未来行为预测，

地面的检测，可以把地面分成大块（不是小块tile），或者在上面覆盖一个大块的透明对象，整体增加一个**对角色的碰撞检测**，如果发生碰撞，则把角色往回拉（g），或者给予一个反向的g

角色下落用统一的重力控制， 跳起后应由重力自然压下去，碰撞检测到地面的时候，给予一个反作用力

游戏循环：控制所有显示对象的动态变化，有定时循环，for循环，桢循环

包括变化和检测

数据结构， 最常用的就是数组，把大量相似对象放在一组

地图常用二维数组,tile地图表示

碰撞检测的几种实现

地图卷动:自动卷 人物行走卷 边缘卷， 中间卷

拖动的实现

巨大地图（容器）和拼接地图，固定的对象和即时对象

Tile地图，对地图（块）的碰撞检测，用数组表示图块不同元素 （ 图块id），分层地图，对象层，纯图片层 ，tile层

将图片某点的坐标，转换为数组索引下标，然后在数组中找到那个块的信息（避免建立太多的显示对象）

世界坐标和 对象在当前窗口的坐标

人物状态机 动画状态机 对于状态复杂的对象，存在状态转换逻辑判断

对象池

对象工厂

大部分对象是矩形包围框，所有可以获取包围框，然后检查矩形之间的碰撞

空间分区， 可以以数组为单位碰撞检测，或者只对同一个分区的对象进行碰撞检测，需要动态添加对象到分区 树结构说不定很合适

更多常用的