23.cocosCreator GUI

对同学们的小提醒

1. 浏览器用google浏览器，不要用别的

1.课前回顾

1.1访问别的节点

1.2节点的常用属性以及方法

1.3计时器

1.4 切换场景

loadScene有个加载场景以及场景的资源过程，所以不要高频率加载切换场景

1.5自定义事件

自定义事件的进化过程

第一个阶段

在creator1.x版本的时候是利用JavaScript typeScript语言自带的事件发送与接收

第二个阶段

cocosCreator在2.x的早期版本推出了内置事件系统

this.node.on this.node.emit 一般是绑定到场景上 或者是导演上

cc.director.emit   cc.director.on

第三个阶段

// 该事件监听每次都会触发，需要手动取消注册
eventTarget.on(type, func, target?);

把eventTarget写成单实例类

1.6作业

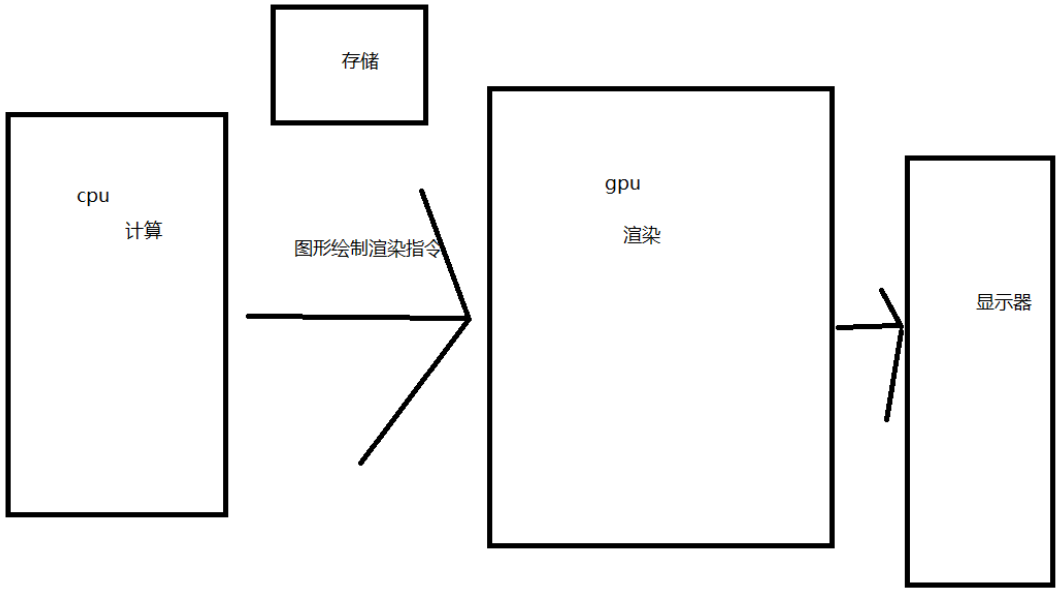
1 运行斗地主游戏 先进入 loading场景 5s后进入大厅场景

2 多脚本事件监听与发送

2.drawCall

在游戏开发中，Draw call 作为一个非常重要的性能指标，直接影响游戏的整体性能表现。Draw call 就是 CPU 调用图形 API，比如 OpenGL，命令 GPU 进行图形绘制。一次 Draw call 就代表一次图形绘制命令，由于 Draw call 带来的 CPU 及 GPU 的渲染状态切换消耗，往往需要通过批次合并来降低 Draw call 的调用次数。批次合并的本质就是在一帧的渲染过程中，保证连续节点的渲染状态一致，将尽可能多的节点数据合并一次性提交，从而减少绘图指令的调用次数，降低图形 API 调用带来的性能消耗，同

时也可以避免 GPU 进行频繁的渲染状态切换。渲染状态就包括：纹理状态，Blend 模式，Stencil 状态，Depth Test 状态等等。



3.Sprite 组件

3.1Sprite 属性

属性	功能说明
Type	渲染模式，包括普通（Simple）、九宫格（Sliced）、平铺（Tiled）和填充（Filled）四种模式，详情请参考下文 **渲染模式** 部分的内容
CustomMaterial	自定义材质
Grayscale	灰度模式，开启后 Sprite 会使用灰度模式渲染。
Color	图片颜色
Sprite Atlas	Sprite 显示图片资源所属的图集
Sprite Frame	渲染 Sprite 使用的 [SpriteFrame 图片资源]
Size Mode	指定 Sprite 的尺寸 **Trimmed** 表示会使用原始图片资源裁剪透明像素后的尺寸 **Raw** 表示会使用原始图片未经裁剪的尺寸 **Custom** 表示会使用自定义尺寸。当用户手动修改过 **Size** 属性后，**Size Mode** 会被自动设置为 **Custom**，除非再次指定为前两种尺寸。
Trim	是否渲染原始图像周围的透明像素区域，详情请参考

添加 Sprite 组件之后，通过从 **资源管理器** 中拖拽 SpriteFrame 类型的资源到 **SpriteFrame** 属性引用中，就可以通过 Sprite 组件显示资源图像。

如果拖拽的 SpriteFrame 资源是包含在一个 Atlas 图集资源中的，那么 Sprite 的 **Atlas** 属性也会被一起设置。

若要动态更换 SpriteFrame 则需要先动态加载图片资源，然后再进行替换，详情请参考

3.2渲染模式

Sprite 组件支持以下几种渲染模式：

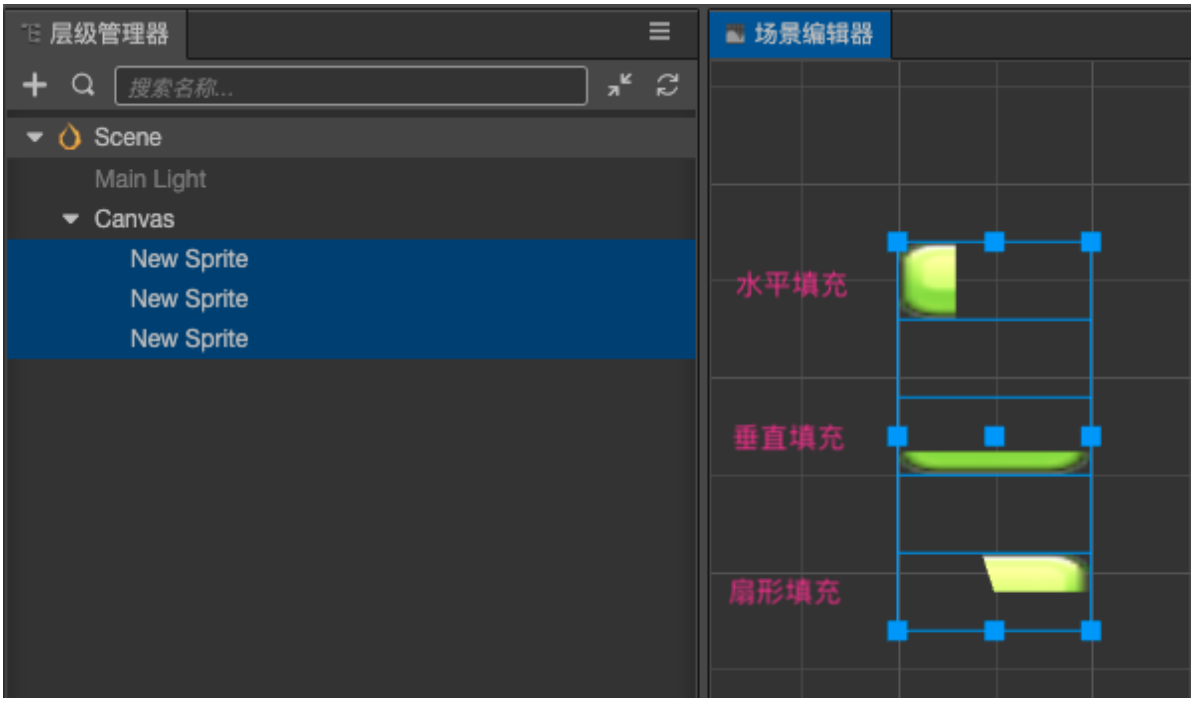
- `普通模式（Simple）`：根据原始图片资源渲染 Sprite，一般在这个模式下我们不会手动修改节点的尺寸，来保证场景中显示的图像和美术人员生产的图片比例一致。
- `九宫格模式（Sliced）`：图像将被分割成九宫格，并按照一定规则进行缩放以适应可随意设置的尺寸(`size`)。通常用于 UI 元素，或将可以无限放大而不影响图像质量的图片制作成九宫格图来节省游戏资源空间。
- `平铺模式（Tiled）`：当 Sprite 的尺寸增大时，图像不会被拉伸，而是会按照原始图片的大小不断重复，就像平铺瓦片一样将原始图片铺满整个 Sprite 规定的大小。
- 填充模式(FILLED):主要用来做游戏进度条

重点讲解九宫格模式和填充模式

3.2.1填充模式(FILLED)

Type 属性选择填充模式后，会出现一组新的属性可供配置：

属性	功能说明
Fill Type	填充类型选择，有 **HORIZONTAL**（横向填充）、**VERTICAL**（纵向填充）和 **RADIAL**（扇形填充）三种。
Fill Start	填充起始位置的标准化数值（从 0 ~ 1，表示填充总量的百分比），选择横向填充时，**Fill Start** 设为 0，就会从图像最左边开始填充
Fill Range	填充范围的标准化数值（同样从 0 ~ 1），设为 1，就会填充最多整个原始图像的范围。
Fill Center	填充中心点，该属性只有选择了 `RADIAL` 填充类型才能修改。决定了扇形填充时会环绕 Sprite 上的哪个点。



3.3Blend

高难度内容 使用地方很少

多图片的混合显示

Creator 编辑器中对部分渲染组件可以进行 Blend Func 的编辑，Src Blend Factor 以及 Dst Blend Factor，分别表示颜色混合时的源颜色和目标颜色的取值模式。颜色混合公式为：

```
FinalRed = (RS * RFactor) + (RD * RFactor)
FinalGreen = (GS * GFactor) + (GD * GFactor)
FinalBlue = (BS * BFactor) + (BD * BFactor)
FinalAlpha = (AS * AFactor) + (AD * AFactor)
```

Constant	RFactor	GFactor	BFactor	AFactor	Description
ZERO	0	0	0	0	将所有颜色乘以 0
ONE	1	1	1	1	将所有颜色乘以 1
SRC_COLOR	RS	GS	BS	AS	将所有颜色乘以源颜色
ONE_MINUS_SRC_COLOR	1 - RS	1 - GS	1 - BS	1 - AS	将所有颜色乘以 1 减去源颜色
DST_COLOR	RD	GD	BD	AD	将所有颜色乘以目标颜色
ONE_MINUS_DST_COLOR	1 - RD	1 - GD	1 - BD	1 - AD	将所有颜色乘以 1 减去目标颜色
SRC_ALPHA	AS	AS	AS	AS	将所有颜色乘以源透明度
ONE_MINUS_SRC_ALPHA	1 - AS	1 - AS	1 - AS	1 - AS	将所有颜色乘以 1 减去源透明度
DST_ALPHA	AD	AD	AD	AD	将所有颜色乘以目标透明度
ONE_MINUS_DST_ALPHA	1 - AD	1 - AD	1 - AD	1 - AD	将所有颜色乘以 1 减去目标透明度

某些情况下，透明图片的边缘部分因为采样到透明区域的背景颜色，会出现黑色边缘的现象，通常为了解决该问题，会设置图片的预乘状态，并设置 Src Blend Factor 为 **ONE**，但是 Blend 模式不一致也会打断批次合并。

Blend 使用的最佳实践

由于 Blend 模式不一致会打断批次合并，所以也应该尽量减少 Blend 模式的改变。比如 PNG 图片的黑边问题，可以在图集打包时进行扩边，而不是用切换 Blend Func 的方式。如果某些表现效果需要特殊的设置，也应该在设定 UI 布局层级时，尽量保证 Blend 模式一致的节点是在同一层级且连续，避免不同 Blend 模式的节点交叉布局，这样即可有效减少 Blend 模式切换导致的 Draw call 增加。

4.Lable

Label 组件用来显示一段文字，文字可以是系统字体，TrueType 字体、BMFont 字体或艺术数字。另外，Label 还具有排版功能。

4.1Label 属性

属性	功能说明
CustomMaterial	自定义材质，使用方法参考 自定义材质
Color	文字颜色。
String	文本内容字符串。
HorizontalAlign	文本的水平对齐方式。可选值有 LEFT、CENTER 和 RIGHT。
VerticalAlign	文本的垂直对齐方式。可选值有 TOP、CENTER 和 BOTTOM。
FontSize	文本字体大小。
FontFamily	文字字体名字。在使用系统字体时生效。
LineHeight	文本的行高。
Overflow	文本的排版方式，目前支持 CLAMP、SHRINK 和 RESIZE_HEIGHT。详情见下方的 Label 排版 或者 文字排版。
EnableWrapText	是否开启文本换行。（在排版方式设为 CLAMP、SHRINK 时生效）
Font	指定文本渲染需要的 字体资源。若要使用艺术数字字体，请参考 艺术数字资源 文档进行配置。 如果使用系统字体，则此属性可以为空。
UseSystemFont	布尔值，是否使用系统字体。
CacheMode	文本缓存类型，仅对 **系统字体** 或 **TTF** 字体有效，BMFont 字体无需进行这个优化。包括 **NONE**、**BITMAP**、**CHAR** 三种模式。详情见下方的 文本缓存类型。
IsBold	文字是否加粗，支持系统字体以及部分 TTF 字体。
IsItalic	文字是否倾斜，支持系统字体以及 TTF 字体。
IsUnderline	文字是否加下划线，支持系统字以及 TTF 字体。

4.2Label 排版

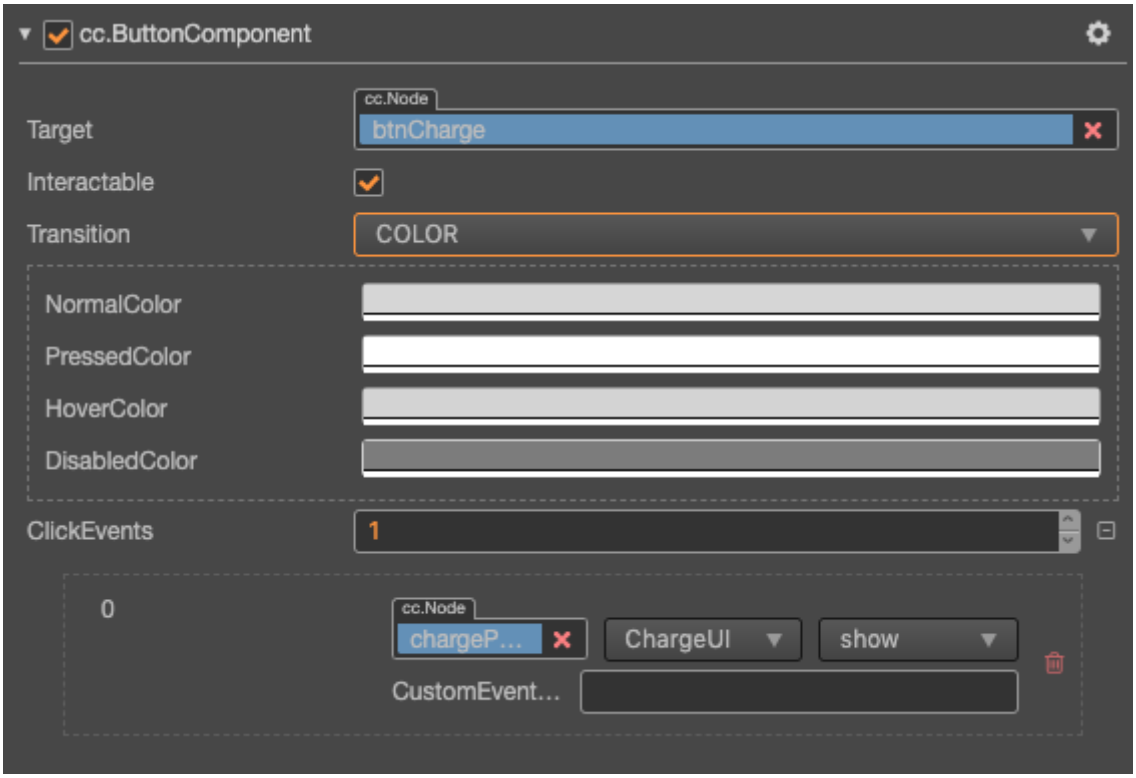
属性	功能说明
CLAMP	文字尺寸不会根据 Content Size 的大小进行缩放。 Wrap Text 关闭的情况下，按照正常文字排列，超出 Content Size 的部分将不会显示。 Wrap Text 开启的情况下，会试图将本行超出范围的文字换行到下一行。如果纵向空间也不够时，也会隐藏无法完整显示的文字。
SHRINK	文字尺寸会根据 Content Size 大小进行自动缩放（不会自动放大，最大显示 Font Size 规定的尺寸）。 Wrap Text 开启时，当宽度不足时会优先将文字换到下一行，如果换行后还无法完整显示，则会将文字进行自动适配 Content Size 的大小。 Wrap Text 关闭时，则直接按照当前文字进行排版，如果超出边界则会进行自动缩放。
RESIZE_HEIGHT	文本的 Content Size 会根据文字排版进行适配，这个状态下用户无法手动修改文本的高度，文本的高度由内部算法自动计算出来。

4.3文本缓存类型（Cache Mode）

类型	功能说明
NONE	默认值，Label 中的整段文本将生成一张位图。
BITMAP	选择后，Label 中的整段文本仍将生成一张位图，但是会尽量参与 动态合图。只要满足动态合图的要求，就会和动态合图中的其它 Sprite 或者 Label 合并 Draw Call。由于动态合图会占用更多内存，**该模式只能用于文本不常更新的 Label**。此模式在节点安排合理的情况下可大幅降低 Draw Call，请酌情选择使用
CHAR	原理类似 BMFont，Label 将以“字”为单位将文本缓存到全局共享的位图中，相同字体样式和字号的每个字符将在全局共享一份缓存。能支持文本的频繁修改，对性能和内存最友好。不过目前该模式还存在如下限制，我们将在后续的版本中进行优化： 1、**该模式只能用于字体样式和字号（通过记录字体的 fontSize、fontFamily、color、outline 为关键信息，以此进行字符的重复使用，其他有使用特殊自定义文本格式的需要注意）固定，并且不会频繁出现巨量未使用过的字符的 Label**。这是为了节约缓存，因为全局共享的位图尺寸为 2048*2048，只有场景切换时才会清除，一旦位图被占满后新出现的字符将无法渲染。 2、Overflow 不支持 SHRINK。 3、不能参与动态合图（同样启用 CHAR 模式的多个 Label 在渲染顺序不被打断的情况下仍然能合并 Draw Call）

5.Button（按钮）组件

Button 组件可以响应用户的点击操作，当用户点击 Button 时，Button 自身会有状态变化。另外，Button 还可以让用户在完成点击操作后响应一个自定义的行为。



点击 **属性检查器** 下面的 **添加组件** 按钮，然后从 **添加 UI 组件** 中选择 **Button**，即可添加 Button 组件到节点上。

5.1Button 属性

属性	功能说明
Target	Node 类型，当 Button 发生 Transition 的时候，会相应地修改 Target 节点的 SpriteFrame，颜色或者 Scale。
Interactable	布尔类型，设为 false 时，则 Button 组件进入禁用状态。
Transition	枚举类型，包括 NONE、COLOR、SPRITE 和 SCALE。每种类型对应不同的 Transition 设置。详情见下方的 **Button Transition** 部分。
Click Event	列表类型，默认为空，用户添加的每一个事件由节点引用，组件名称和一个响应函数组成。详情见下方的 **Button 点击事件** 部分。

6.EditBox

EditBox 是一种文本输入组件，该组件让你可以轻松获取用户输入的文本。

6.1EditBox 属性

属性	功能说明
BackgroundImage	输入框背景节点上挂载的 Sprite 组件对象
FontColor	输入框文本的颜色
FontSize	输入框文本的字体大小
InputFlag	指定输入标识：可以指定输入方式为密码或者单词首字母大写（仅支持 Android 平台）
InputMode	指定输入模式：ANY 表示多行输入，其它都是单行输入，移动平台上还可以指定键盘样式。
LineHeight	输入框文本的行高
MaxLength	输入框最大允许输入的字符个数
Placeholder	输入框占位符的文本内容
PlaceholderFontColor	输入框占位符的文本字体颜色
PlaceholderFontSize	输入框占位符的文本字体大小
PlaceholderLabel	输入框占位符节点上挂载的 Label 组件对象
ReturnType	指定移动设备上面回车按钮的样式
String	输入框的初始输入内容，如果为空则会显示占位符的文本
TabIndex	修改 DOM 输入元素的 tabIndex，这个属性只有在 Web 上面修改有意义。
TextLabel	输入框输入文本节点上挂载的 Label 组件对象

7.Layout 组件

Layout 是一种容器组件，容器能够开启自动布局功能，自动按照规范排列所有子物体，方便用户制作列表、翻页等功能。

7.1Layout 属性

属性	功能说明
Type	布局类型，支持 **NONE**、**HORIZONTAL**、**VERTICAL** 和 **GRID**。
ResizeMode	缩放模式，支持 **NONE**、**CHILDREN** 和 **CONTAINER**。
PaddingLeft	排版时，子物体相对于容器左边框的距离。
PaddingRight	排版时，子物体相对于容器右边框的距离。
PaddingTop	排版时，子物体相对于容器上边框的距离。
PaddingBottom	排版时，子物体相对于容器下边框的距离。
SpacingX	水平排版时，子物体与子物体在水平方向上的间距。NONE 模式无此属性。
SpacingY	垂直排版时，子物体与子物体在垂直方向上的间距。NONE 模式无此属性。
HorizontalDirection	指定水平排版时，第一个子节点从容器的左边还是右边开始布局。当容器为 `GRID` 类型时，此属性和 `Start Axis` 属性一起决定 Grid 布局元素的起始水平排列方向。
VerticalDirection	指定垂直排版时，第一个子节点从容器的上面还是下面开始布局。当容器为 `GRID` 类型时，此属性和 `Start Axis` 属性一起决定 Grid 布局元素的起始垂直排列方向。
CellSize	此属性只在 `GRID` 布局、`Children` 缩放模式时存在，指定网格容器里面排版元素的大小。
StartAxis	此属性只在 `GRID` 布局时存在，指定网格容器里面元素排版指定的起始方向轴。
AffectedByScale	子节点的缩放是否影响布局。
AutoAlignment	自动对齐，在 `Type` 类型为 **HORIZONTAL** 或 **VERTICAL** 模式下，保证另外一个轴向值始终为 0。
Constraint	布局约束，可以在某个方向上约束排列数量，支持 **NONE**、**FIXED_ROW** 和 **FIXED_COL**。
ConstraintNum	布局约束值，在 `Constraint` 的类型为 **FIXED_ROW** 或 **FIXED_COL** 模式下有效。

8.Widget 组件参考

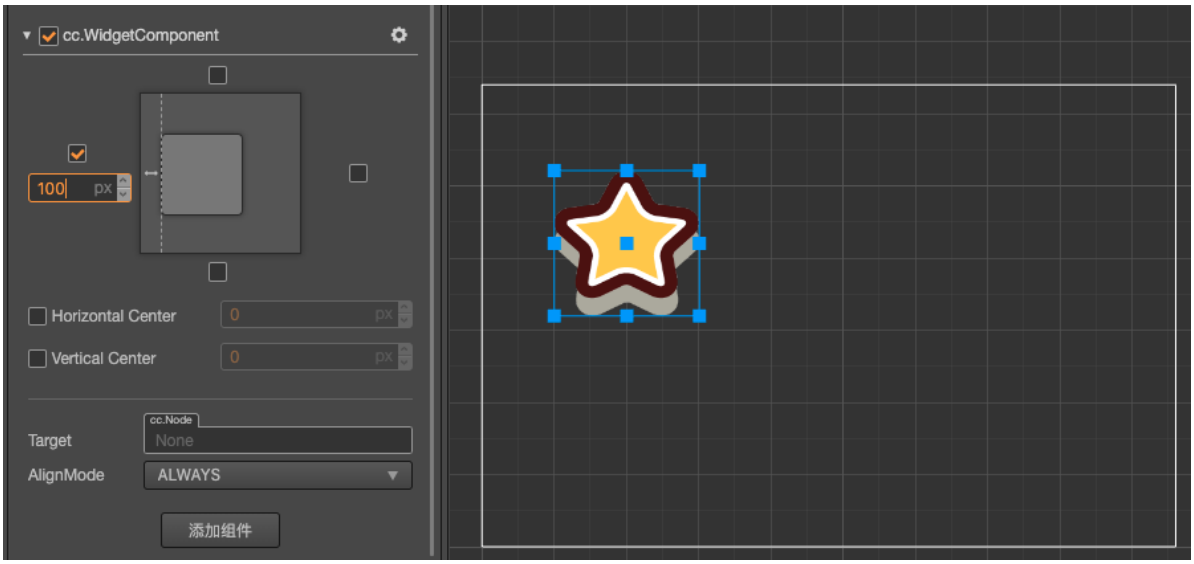
8.1选项

选项	说明	备注
Top	对齐上边界	选中后，将在旁边显示一个输入框，用于设定当前节点的上边界和父物体的上边界之间的距离。
Bottom	对齐下边界	选中后，将在旁边显示一个输入框，用于设定当前节点的下边界和父物体的下边界之间的距离。
Left	对齐左边界	选中后，将在旁边显示一个输入框，用于设定当前节点的左边界和父物体的左边界之间的距离。
Right	对齐右边界	选中后，将在旁边显示一个输入框，用于设定当前节点的右边界和父物体的右边界之间的距离。
HorizontalCenter	水平方向居中	
VerticalCenter	竖直方向居中	
Target	对齐目标	指定对齐参照的节点，当这里未指定目标时会使用直接父级节点作为对齐目标
AlignMode	指定 Widget 的对齐方式，用于决定运行时 Widget 应何时更新	通常设置为 **ALWAYS**，每次节点产生变动时重新对齐。 设置为 **ONCE** 时，仅在组件初始化时进行一次对齐。**ON_WINDOW_RESIZE** 时会在每次窗口变动时候更新一次

8.2对齐边界

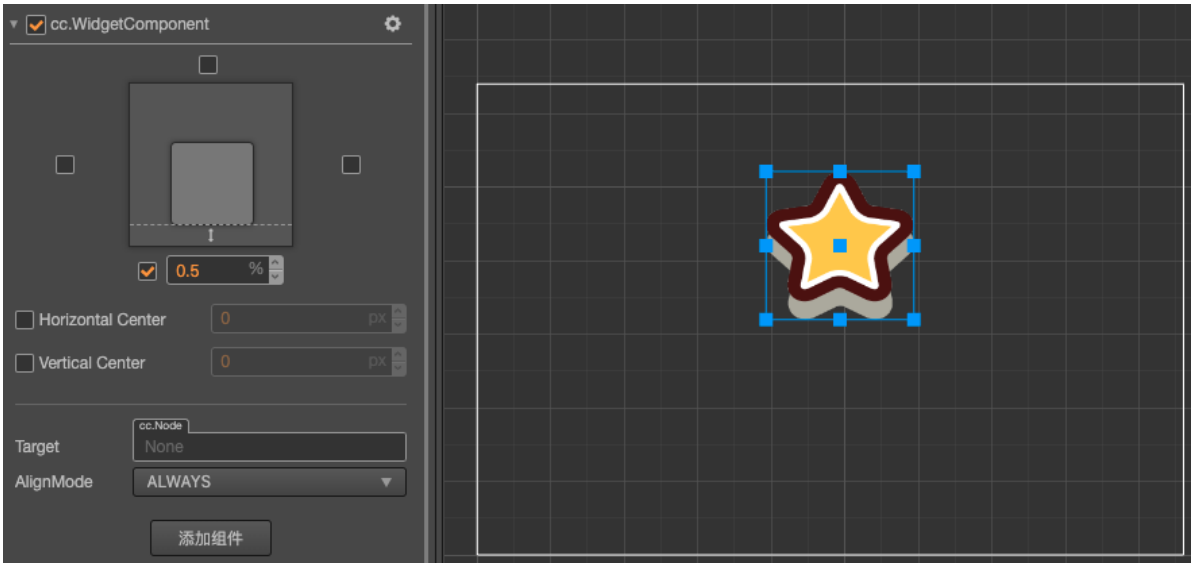
我们可以在 Canvas 下新建一个 sprite，在 sprite 节点上添加一个 Widget 组件，然后做如下一些测试：

左对齐，左边界距离 100 px



下对齐，下边界距离 50%

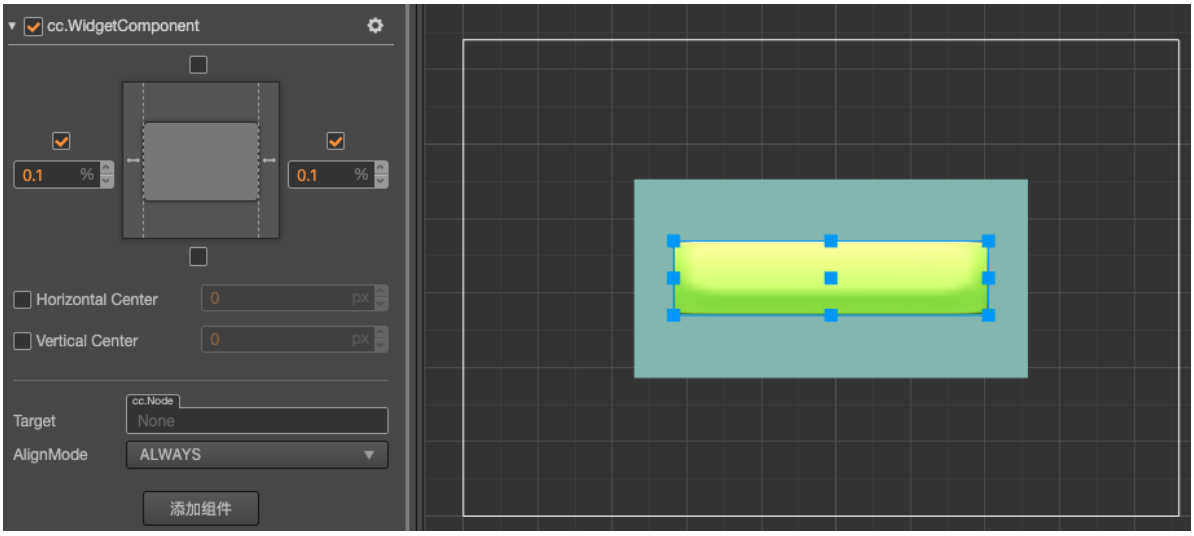
百分比将以父节点的宽或高作为基准。



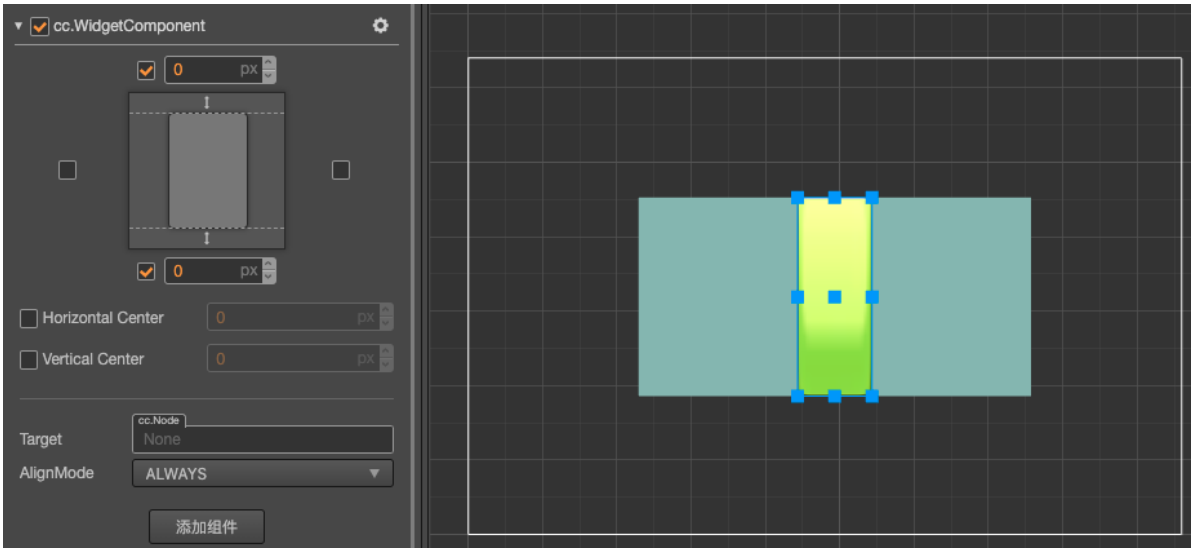
8.3约束尺寸

如果左右同时对齐，或者上下同时对齐，那么在相应方向上的尺寸就会被拉伸。 下面演示一下，在场景中放置两个矩形 Sprite，大的作为对话框背景，小的作为对话框上的按钮。按钮节点作为对话框的子节点，并且按钮设置成 Sliced 模式以便展示拉伸效果。

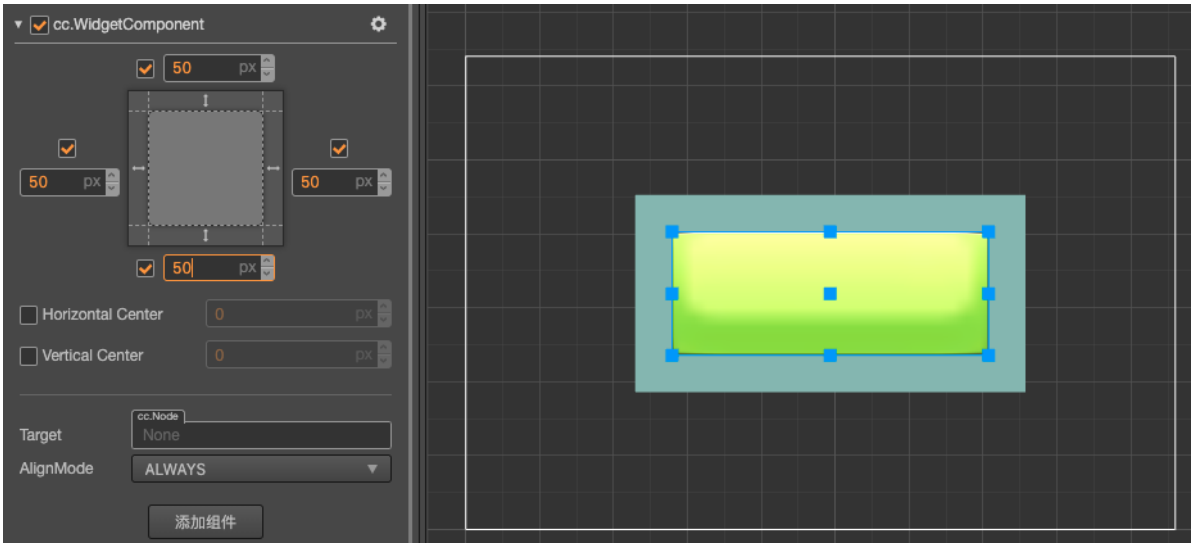
宽度拉伸，左右边距 10%



高度拉伸，上下边距 0，同时水平居中



水平和竖直同时拉伸，边距 50 px



9.作业

1.整理笔记

2.重新编辑游戏loading场景与大厅场景