CocoStudio GUI 库使用教程

编辑器以及源代码的下载:
http://bbs.cocostudio.org/forum.php?mod=viewthread&tid=692#lastpost

http://www.cocostudio.org/detail.php?id=3754

工程的搭建流程请参考视频教程: http://www.youku.com/playlist_show/id_19206519.html

1.UI 框架的使用方法以及常用接口介绍

UI 系统的初始化:

通过接口 COCOUISYSTEM->resetSystem(cocos2d::CCNode container);

是通过接口 COCOUISYSTEM->replaceUISceneWithFile(
cocos2d::CCNode container, const char* filename, int fileType, bool enableAdapt, bool scaleAdapt, bool equalProportions, int priority);

ui 层的渲染是需要一个 CCNode 作为渲染容器的,所有的 ui 控件全部渲染在一 个 CCNode(或 CClayer,CCScene 等等)之上,所以在初始化 ui 系统时,需要 给 ui 系统传入一个 CCNode 参数作为渲染容器。

两个接口的不同之处在于:

是纯粹的初始化 ui 系统,它会清除当前 ui 系统中所有的控件,并重新初始 化,以参数 ccnode 作为渲染容器。

是通过一个 json 文件(由 ui 编辑器导出生成)来初始化一个 ui 系统,ui 系统 同样会清除当前 ui 系统中的控件,并初始化成 json 文件中描述的 ui 控件。 参数中,fileType 是指文件的类型,0 为 plist,1 为 json,通常我们使用 json 文件,所以参数传入1, enableAdapt 参数是指这个文件创建出来的 ui 场 景是否需要适配到当前设备的分辨率,scaleAdapt 是指适配分辨率时,是否需 要对单个控件进行尺寸缩放适配,equalProportions 是指尺寸缩放的适配是否 需要等比缩放,最后一个参数是 ui 系统的点击事件优先级,如同 CCLayer 的点 击事件接收的优先级,其默认值为-1,表示最优先接收点击事件,同样我们可 以通过 COCOUISYSTEM->setPriority(int);接口来设置 ui 系统的点击优先级。

UI 控件的创建和使用及注意事项:

UI 控件的创建方法:

UI 控件(CocoWidget)的创建方法有 2 种。

一种是通过编码的方式创建一个 ui 控件,并通过编码的形式赋予控件属性。

```
59      cs::CocoButton* button = cs::CocoButton::create();
60      button->setPosition(ccp(100, 100));
61      button->setBeTouchAble(true);
62      button->setTextures("normal.png", "pressed.png", "disabled.png");
```

如上,通过 Create()静态函数来构造一个按钮(CocoButton),这里需要注意的 是,这个 Create()和 CCNode::Create()不同,通过 Create()方法创建出来的 ui 控件(CocoWidget)没有使用 autorelease 机制,内存管理在 UI 系统中已经 做了管理,关于 ui 控件的删除和内存释放,我们后面会讲解。 第二行,给按钮设置位置,第三行,设置按钮为可点击,第四行,给按钮设置 三种状态下的图片(正常,按下,不可用)。

一种是通过读取编辑器导出的文件来创建一个 ui 控件,通过 UI 系统的接口来完成。

```
64      cs::CocoWidget* widget = COCOUISYSTEM->createWidgetFromFile_json("ui.Exportjson");
```

仅需要一行代码,非常简单,创建好的 widget,属性也已经被赋值,这里需要 注意的是,” ui.Exportjson” 文件中,描述的是一个控件的类型,属性,以及结 构树,也就是说,json 文件中描述的是一棵 ui 树,它可能有孩子。如图:

这里需要注意的是,有些同学会有疑问,我如何把一个创建好的 UI 控件添加到 游戏中(CCLayer,CCScene)呢?

我们上面提到了,我们有个 UI 系统 (COCOUISYSTEM),我们创建好的 ui 控件是被直接添加到 UI 系统中的,因为我们的 UI 系统和一个渲染容器(CCNode,CCLayer,etc.)做了关联(COCOUISYSTEM->resetSystem(cocos2d::CCNode)) ,所以不需要关心我们如何把一个 UI 控件添 加到一个 CCLayer 中,我们只需要把它添加到 UI 系统中就好,如:COCOUISYSTEM->addWidget(CocoWidget);,当然,我们的建议做法是创建一个CocoPanel,作为当前 ui 的底层,通过COCOUISYSTEM->addWidget(panel);, 将其添加到 UI 系统中,然后我们只需要就所有创建的 ui 控件添加到这个底层 上就可以了,这样结构看起来更清晰(panel->addChild(CocoButton));

2.各个控件的使用方法:

下面,我们逐一介绍每个控件的使用方法以及特性:

CocoButton

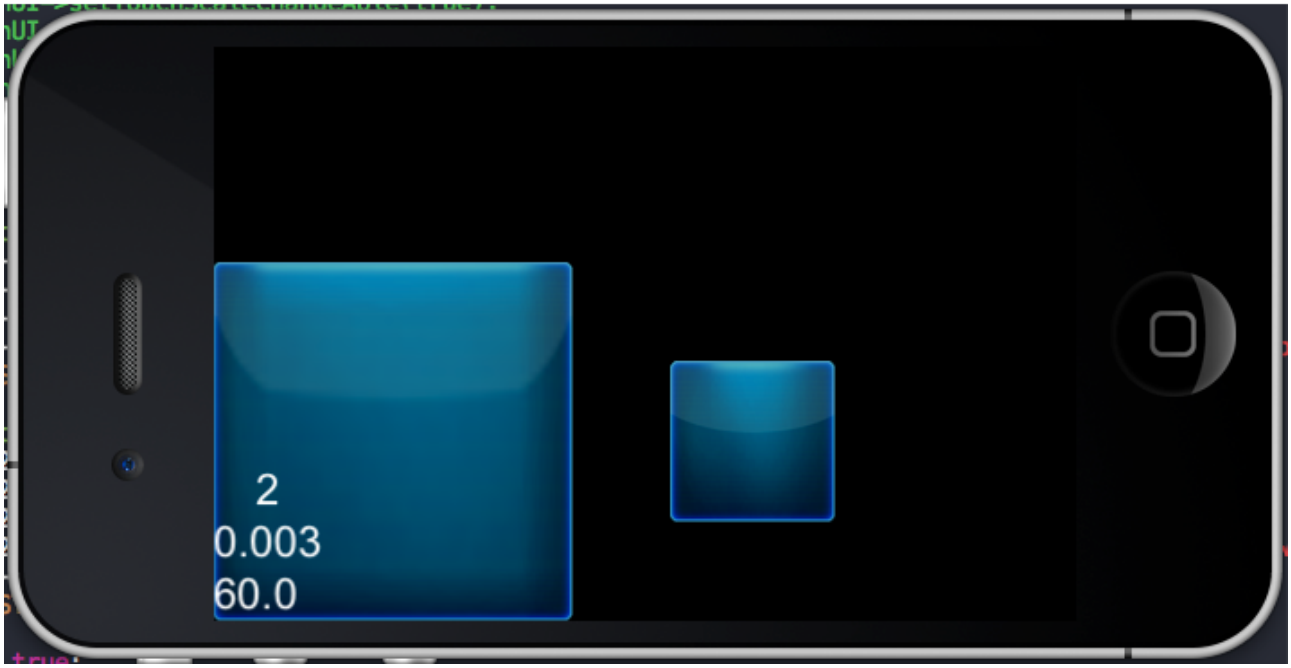
CocoButton 是 ui 中常见的按钮:

创建方式如图:

```
59      cs::CocoButton* button = cs::CocoButton::create();
60      button->setPosition(ccp(100, 100));
61      button->setBeTouchAble(true);
62      button->setTextures("normal.png", "pressed.png", "disabled.png");
```

我们需要用三张图片来表示按钮的三种不同状态下的样子,normal,pressed, disabled。

特性:CocoButton 支持九宫格的图片来作为按钮的三种状态下的图片,什么是九宫格? 如何创建一个九宫格按钮? 九宫格图片是指一张尺寸非常小,可通过九宫格方式拉伸为任意尺寸的图片, 这样做既节省资源,又不影响视觉效果。



上图中,左边为设置九宫格拉伸的效果,右边为原图效果。
创建一个具有九宫格拉伸效果的按钮非常简单:

```
59      cs::CocoButton* button = cs::CocoButton::create();
60      button->setPosition(ccp(100, 100));
61      button->setBeTouchAble(true);
62
63      button->setScale9Enable(true);
64      button->setTexturesScale9("scale9normal.png", "scale9pressed.png", "scale9disabled.png",CCRectZero);
65
66      COCOUISYSTEM->getCurScene()->addWidget(button);
```

请注意红框中的两行代码,button->setScale9Enable(true),这是表示开启按 钮的九宫格功能,默认是不开启的,设置按钮的九宫格功能开启后,我们需要 调用 button->setTexturesScale9()来为其设置图片,(最后一个参数为九宫 格图片的内矩形,即不可拉伸的部分,如果传入零矩形,则表示启用默认值, 内矩形为宽和高的各三分之一),而不是之前的button->setTextures(),这 里需要注意的是,如果我们一开始使用 button->setTextures()为一个没有开 启九宫格功能的按钮设置了图片,然后又调用 button->setScale9Enable(true),按钮将会清除掉之前的图片,这时我们需要调用setTexturesScale9()来为按钮重新设置图片,同样,我们给一个已经设置九宫 格功能开启的按钮设置一个 setScale9Enable(false),按钮也会清除掉九宫格 图片,我们需要调用setTextures()接口来为按钮重新设置图片。

CocoTextButton:

CocoTextButton 是文本按钮,从 CocoButton 继承而来,比 CocoButton多了一 个文本显示。

创建方式如下:

```
124     cs::CocoTextButton* tb = cs::CocoTextButton::create();
125     tb->setTextures("normal.png", "pressed.png", "disabled.png");
126     tb->setText("textbutton");
127     tb->setFontName("宋体");
128     tb->setFontSize(10);
```

特性:
可通过上图接口为按钮上的文本设置属性。
文本按钮同样支持九宫格功能。

CocoCheckBox:

CocoCheckBox 是复选框,用于纪录状态的控件。

创建方式如下:

```
76      cs::CocoCheckBox* checkBox = cs::CocoCheckBox::create();
77      checkBox->setPosition(ccp(100, 100));
78      checkBox->setTextures("backGround.png", "backGroundSelected.png", "cross.png", "backGroundDisabled.png",
            "crossDisabled.png");
79      checkBox->setSelectedState(true);
```

CocoCheckBox 需要 5 张图片来完成渲染,背景,背景被选中,勾,背景不可用, 勾不可用。

特性:我们可以通过 checkBox->setSelectedState(bool);来设置一个复选框的初始 选中状态。

CocoImageView:

CocoImageView 是图片控件,用于显示一张纹理。

创建方式如下:

```
81      cs::CocoImageView* iv = cs::CocoImageView::create();
82      iv->setPosition(ccp(100, 100));
83      iv->setTexture("image.png");
```

通过一个文件名称来为图片控件添加纹理。 (注意:很多同学会疑问,CocoWidget 如何添加一个 CCSprite?其实使用 CCSprite 的目的是渲染图片,完全可以用 CocoImageView 来渲染)

特性:九宫格,和按钮一样,图片也支持九宫格,用法和按钮一致,iv- >setScale9Enable(bool),注意事项也和按钮一致。

CocoLabel:

CocoLabel 是文本标签,通常用语表示一行文字。

创建方式如下:

```
85      cs::CocoLabel* label = cs::CocoLabel::create();
86      label->setPosition(ccp(100, 22));
87      label->setText("I am a Label");
88      label->setFontName("宋体");
89      label->setFontSize(20);
90      label->setColor(200, 244, 123);
91      std::string str = label->getStringValue();
```

特性:

内容:通过 setText()函数来设置文本内容，通过 getStringValue()来获取文本内容。

字体:通过 setFontName()函数来设置文本需要的字体，通过 setFontSize()函数来设置文本的字号。

颜色:通过 setColor()函数来设置文本颜色。

CocoTextArea:

CocoTextArea 是文本区,用于显示文本,于 CocoLabel 不同的是,label 只能 显示一行文字,而 CocoTextArea 可以将文字显示在一个区域内,并可以换行。 创建方式如下:

```
115     cs::CocoTextArea* ta = cs::CocoTextArea::create();
116     ta->setPosition(ccp(100, 100));
117     ta->setTextAreaSize(100, 100);
118     ta->setTextColor(0, 0, 0);
119     ta->setTextHorizontalAlignment(0);
120     ta->setTextVerticalAlignment(0);
121     ta->setFontName("宋体");
122     ta->setFontSize(20);
```

特性:通过 setTextAreaSize()函数来设置文本区的大小。通过 setTextHorizontalAlignment()来设置文字在文本区中的横向停靠(左、 右、居中)。

```
typedef enum
{
    kCCTextAlignmentLeft,
    kCCTextAlignmentCenter,
    kCCTextAlignmentRight,
} CCTextAlignment;
```

通过 setTextVerticalAlignment()函数来设置文字在文本区中的竖向停靠(上、 下、居中)。

```
typedef enum
{
    kCCVerticalTextAlignmentTop,
    kCCVerticalTextAlignmentCenter,
    kCCVerticalTextAlignmentBottom,
} CCVerticalTextAlignment;
```

CocoLabelAtlas:

CocoLabelAtlas 是数字标签,通常使用是用一张纹理来渲染一些数字,减少渲 染和内存消耗。

创建方式如下:

```
92      cs::CocoLabelAtlas* labelAtlas = cs::CocoLabelAtlas::create();
93      labelAtlas->setPosition(ccp(100, 10));
94      labelAtlas->setProperty("1234.5", "labelatlas.png", 20, 20, "0");
95      labelAtlas->setStringValue("100.3");
```

通过 setProperty 函数来为数字标签设置属性,第一个参数为数字内容,第二个 为数字标签用到的图片,通常为这个样子:

第三个参数为单个数字的宽(像素),第四个参数为单个数字的高(像素),第五 个参数为图片中,起始字符为什么,如上图,起始字符我们需要填写”0”。

特性:我们可以通过 setStringValue 函数来为数字标签设置内容。

CocoLoadingBar:

CocoLoadingBar 是进度条,通常用于做进度显示使用。

创建方式如下:

```
97      cs::CocoLoadingBar* loadingBar = cs::CocoLoadingBar::create();
98      loadingBar->setTexture("loadingbar.png");
99      loadingBar->setPosition(ccp(100, 100));
100     loadingBar->setDirection(0);
101     loadingBar->setPercent(100);
```

进度条需要一张纹理作为渲染。

特性:通过 setDirction()函数来设置进度条的朝向,0 为从左到右,1 为从右到左。通过 setPercent()函数来设置进度条的显示比例,0~100。

CocoSlider:

CocoSlider 是滑动条,通常用于做一些值的控制,如音量等。

创建方式如下:

```
103     cs::CocoSlider* slider = cs::CocoSlider::create();
104     slider->setBarTextureScale9Enable(true);
105     slider->setBarTextureScale9("sliderbarscale9.png", 0, 0, 0, 0);
106     slider->setBarLength(100);
107     slider->setSlidBallTextures("normal.png", "pressed.png", "disabled.png");
108
109     slider->setBarTextureScale9Enable(false);
110     slider->setBarTexture("sliderbar.png");
111
112     slider->setSlidBallPercent(100);
```

滑动条需要 4 张图片作为渲染,滑动按钮的三个状态下图片,滑动条图片。

特性:九宫格: 滑动条的“条” 可设置为九宫格,如果开启九宫格,则需要设置滑动条的长度, 否则滑动条的长度默认为九宫格图片的长度。如果未开启九宫格功能,则设置 滑动条长度无效。

设置滑动条百分比:通过 setSlidBallPercent()函数来设置滑动按钮所在滑动条上的百分比(位置)。

CocoTextField:

CocoTextField 是输入框,通过输入框来输入文字。

创建方式如下:

```
131     cs::CocoTextField* tf = cs::CocoTextField::create();
132     tf->setPlaceHolder("input words");
133     tf->setFontSize(10);
134     tf->setText("aaaa");
135     tf->getStringValue();
```

特性:通过 setPlaceHolder()函数来设置输入框的 placeholder。 通过 getStringValue()来获取输入的文字。

CocoPanel:

CocoPanel 是层容器,用于装各种控件,组成一个复杂面板,不同于其他控件 的是,只有容器添加子节点是合理的。

创建方式如下:

```
137     cs::CocoPanel* panel = cs::CocoPanel::create();
138     panel->setBackGroundImageScale9Enable(true);
139     panel->setSize(100, 100);
140     panel->setBackGroundImageScale9("background.png", CCRectZero);
141     panel->setClipAble(true);
```

特性:
九宫格:
层容器的背景图支持九宫格,注意事项、用法和按钮类似,如上图接口。
裁切:

容器是可设置裁切的,默认不开启,开启裁切后,容器会裁切掉自己范围以外 的子节点的渲染,如上图,panel 的尺寸为 100,100,开启裁切,如果它有一个 子节点的位置是 200,200,那么这个子节点将不被渲染。

CocoScrollView:

CocoScrollView 是滚动层,从层容器继承而来,具有层容器所有特性(背景, 裁切)。同时 CocoScrollView 中的子节可在 CocoScrollView 中滚动。

创建方式如下:

```
144     cs::CocoScrollView* sc = cs::CocoScrollView::create();
145     sc->setBackGroundImage("bg.png");
146     sc->setClipAble(true);
147     sc->setUpdateEnable(true);
148     sc->setDirection(0);
```

特性:同 CocoPanel 的特性。通过 setDirection()函数来设置滚动层的滚动方向,0 为竖向,1 为横向。

注意,使用滚动层时,需要调用 setUpdateEnable(true),否则不会有滚动缓 冲的效果。

还需注意的是,往滚动层中添加子节点时,子节点的坐标是存在于滚动层的坐 标系下,所以我们需要给子节点设置合理的坐标,如子 1 设置为(0,0),子 2 设置 为(0,50),子 3 设置为(0,100),等等,根据具体需求来设置子节点。

仍需注意的是,当子节点的总尺寸(所有子节点的外接矩形尺寸),不能填充满 滚动层的尺寸时,滚动层会将子节点进行停靠,上下左右或居中,而不会发生 滚动。通过 setBerthOrientation()函数来设置停靠方向。

```
enum SCROLLVIEW_BERTH_ORI
{
    SCROLLVIEW_BERTH_ORI_NONE,
    SCROLLVIEW_BERTH_ORI_TOP,
    SCROLLVIEW_BERTH_ORI_BOTTOM,
    SCROLLVIEW_BERTH_ORI_VERTICAL_CENTER,
    SCROLLVIEW_BERTH_ORI_LEFT,
    SCROLLVIEW_BERTH_ORI_RIGHT,
    SCROLLVIEW_BERTH_ORI_HORIZONTAL_CENTER
};
```

3.UI 控件的事件添加:

ZOrder 的设置:Ui 控件的点击事件响应是根 据 ui 控件的 ZOrder 来进行排 序的,通过

CocoWidge->setWidgetZOrder(int)函数来设 置

控件的 z 顺序,这个 z 顺序会同时影响到控 件的渲染层级

和事件响应层级,也就是说,只要控件渲染在 上层,则其

一定会被优先点击。

UI 控件添加事件,所有控件(CocoWidget)都有通用的事件:按下, 滑动,抬起,取消点击。在我们为控件添加事件之前,首先要确保 控件是可点击的,调用 cocowidget->setBeTouchAble(true);如果 你是通过编辑器文件创建的控件,请选择编辑器中控件的“交互” 为勾选状态。

```
59      cs::CocoButton* button = cs::CocoButton::create();
60      button->setPosition(ccp(100, 100));
61      button->setBeTouchAble(true);
62      COCOUISYSTEM->getCurScene()->addWidget(button);
63      button->addPushDownEvent(this, coco_pushselector(HelloWorld::menuCloseCallback));
64      button->addMoveEvent(this, coco_moveselector(HelloWorld::menuCloseCallback));
65      button->addReleaseEvent(this, coco_releaseselector(HelloWorld::menuCloseCallback));
66
```

还有一些特殊控件的特殊事件:

Slider: 百分比发生变化时会触发的事件:

```
104     cs::CocoSlider* slider = cs::CocoSlider::create();
105     slider->setBarTextureScale9Enable(true);
106     slider->setBarTextureScale9("sliderbarscale9.png", 0, 0, 0, 0);
107     slider->setBarLength(100);
108     slider->setSlidBallTextures("normal.png", "pressed.png", "disabled.png");
109     slider->addPercentChangedEvent(this, coco_percentchangedselector(HelloWorld::init));
```

我们一般是在 release 的事件触发时,处理我们自己的游戏逻辑, 注意,请不要出现以下情况:

```
151     {
152     cs::CocoButton* button = cs::CocoButton::create();
153     button->setPosition(ccp(100, 100));
154     button->setBeTouchAble(true);
155     COCOUISYSTEM->getCurScene()->addWidget(button);
156     button->addPushDownEvent(this, coco_pushselector(HelloWorld::menuCloseCallback));
157     }
158     return true;
159 }
160
161 void HelloWorld::menuCloseCallback(CCObject* pSender)
162 {
163     ((cs::CocoButton*)pSender)->removeFromParentAndCleanup(true);
164 }
```

在按钮 pushdown 事件中,做删除该按钮的操作,因为 pushdown 后, 随后还会右 releaseup 的事件被触发,但是此时控件以被删,则会 导致程序崩溃。

4.UI 控件的查找、删除和内存释放:

查找: 每个控件都有自己的名字,建议每个控件使用不同的名字或 tag。

```
159    COCOUISYSTEM->getWidgetByName("button");
```

```
159    COCOUISYSTEM->getWidgetByTag(100);
```

这样的操作是在整个 ui 系统中查找名称为 button(或 tag 为 100)的 控件,全局搜索,效率会相对低些。 如果你明确的知道,要找的控件是在哪个节点下,可以通过以下方式:

```
9    COCOUISYSTEM->checkWidgetByName(root, "button");
```

已知 button 会在 root 下面,那么就可以这样查找,效率会相对高 些。

删除和内存释放:

如果想要删除整个系统的 ui,则可以调用 COCOUISYSTEM- >resetSystem()或调用 COCOUISYSTEM->cleanUIScene().

关于场景切换,建议的做法是在即将退出的场景(CCScene)的 onExit 中调用 COCOUISYSTEM->cleanUIScene(),在即将进入的场景 的 onEnter 中调用 COCOUISYSTEM->resetSystem().即退出即释放, 进入即加载(目前没有做 ui 的缓存),否则将会出现 ui 事件响应 出现问题,因为 ui 系统需要重新初始化事件响应系统。

前边提到,我们不需要管理内存,想要删除一个 ui 控件,只需要调 用 removeFromParentAndCleanUp 接口就可以删除该控件。

```
button->removeFromParentAndCleanup(true);
```

参数若为 true,则表示彻底删除,该控件将销毁,包括其身上的子 节点,并释放内存。若参数为 false,则表示不彻底删除,该节点可保留,用作别处。

5.UI 动画的使用:

ui 动画的使用非常简单,只需要创建我们需要的 CCAction,然后调 用 CocoWidget->runAction();即可,如下图:

```
    cs::CocoWidget* widget = (cs::CocoWidget*)pSender;
    widget->stopAllActions();
    widget->setColor(0, 0, 255);
    CCTintTo* tt = CCTintTo::create(0.5f, 0, 255, 0);
    CCTintTo* tt2 = CCTintTo::create(0.5f, 255, 255, 255);
    CCFadeOut* fo = CCFadeOut::create(0.5f);
    CCFadeIn* fi = CCFadeIn::create(0.5f);
    CCSequence* seq = CCSequence::create(CCMoveTo::create(1.0, ccp(300, 300)),CCScaleTo::create(1, 0.5),CCMoveTo::create
        (1.0, ccp(127, 96)),CCScaleTo::create(1.0, 1.0),CCRotateTo::create(0.5, 720),CCRotateTo::create(0.5, 0),tt,tt2,fo
        ,fi,NULL);
    widget->runAction(seq);
```

6.UI 控件状态控制:

属性控制:CocoWidget 包含了所有 CCNode 的属性设置,可以随意的设置一个 控件的位置,缩放,旋转,颜色,透明度等等。如下图:

```
    cs::CocoButton* button = cs::CocoButton::create();
    button->setPosition(ccp(100, 100));
    button->setColor(200, 200, 200);
    button->setOpacity(100);
    button->setScale(100);
    button->setScaleX(100);
    button->setScaleY(100);
    button->setRotation(100);
```

状态控制:我们可以设置控件的状态,当前有 2 种状态:激活(active)和不可 用(diabled),我们可以通过调用 CocoWidget 的 active()和 disable()来控制控件是否可用。

```
    cs::CocoButton* button = cs::CocoButton::create();
    button->disable();
    button->active();
```

7.多分辨率适配:

分辨率适配可以通过接口

```
COCOUISYSTEM->replaceUISceneWithFile(this, "UIRES/CocoGUISample.json", 1, true,true,true);
```

来完成,参数的意义在前面已经介绍过。如果想动态创建 ui 控件,并完成适配,可以通过下面这个接口:

```
COCOUISYSTEM->createWidgetFromFileWithAdapt_json("UIRES/CocoGUISample.json", true, true);
```

使用这个接口动态创建出来一个已经适配好的控件树,直接添加到场景中。

8.纹理 cache 的使用:

纹理缓存,是节约内存,提高效率的有效途径,当我们为 一个按钮设置图片时,可以使用缓存中的纹理:

```
61      cs::CocoButton* button = cs::CocoButton::create();
62      button->setTextures("nomal.png", "pressed.png", "disabled.png",true);
```

最后一个参数 默认参数为 false,如果我们传入 true,则 表示 “normal.png”这几张图是需要从cache中读取,那么 如何添加纹理缓存呢?UI 系统提供了这样的接口:

```
64      //添加纹理缓存
65      COCOUISYSTEM->addSpriteFrame("Image.plist");
66      //释放纹理缓存
67      COCOUISYSTEM->removeSpriteFrame("Image.plist");
68      //释放所有缓存
69      COCOUISYSTEM->removeAllSpriteFrame();
```