### 4.2 使用Cocos Studio

本节学习使用Cocos Studio来完成一些原来比较繁琐的工作，比如设计控件或是实现某种场景。

Cocos Studio是由触控科技引擎团队研发的一款基于Cocos2d-x的免费游戏开发的工具集，其包含了游戏开发中核心的几个游戏编辑器：UI编辑器、动画编辑器、场景编辑器、数据编辑器，它们用于处理游戏中的动画资源、UI界面、游戏场景、游戏数据。Cocos Studio针对开发团队中不同的职业进行深度设计，规范了整个开发流程，让开发团

队中每个人各执其职，发挥自己最大的作用，从而实现高效率、高质量、低风险、低成本。

【本章知识点】

* Cocos Studio编辑器的使用
* 使用Cocos Studio实现APP界面
* 使用Cocos Studio自定义列表控件
* Cocos Studio中骨骼编辑器的使用

1. **Cocos Studio的作用**

使用Cocos创建的项目，初始场景中有如下代码：

auto rootNode = CSLoader::createNode("MainScene.csb");

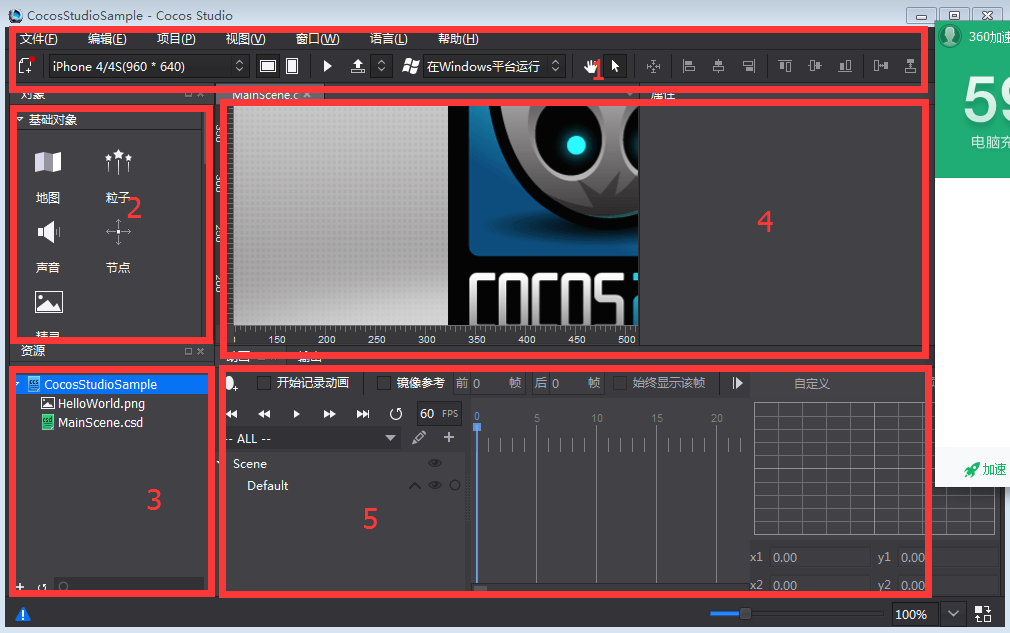
addChild(rootNode);

上面两条语句的作用就是加载了一个名为MainScene.csb的资源，并显示在场景中，而这里的这个csb格式的资源文件就是在Cocos Studio中生成的。

在Delphi、C#和VB中都可以通过拖拽控件的方式来实现一个比较复杂的界面，即使用标准控件会让界面看起来比较丑，但却十分方便。在安卓开发中则可以通过拖拽的方式生成XML布局文件，然后在响应的类中直接调用。

在Cocos Studio中也可以实现类似的功能来解放开发者，但是仅仅是类似而不是相同。因为在游戏开发中需要更加复杂的控件和非常华丽的动画，而作为跨平台的开发引擎辅助工具，需要考虑到对碎片化的移动设备进行适配，因此Cocos Studio要远比哪些常规的UI编辑器强大得多。

**（2）Cocos Studio中各部分的功能**



标注1的部分是在Cocos Studio中常用的一些按钮以及菜单，比如运行设计好的界面或者调整屏幕分辨率等。

标注2的部分是一些常用的控件，比如一个按钮或是一行文字。当然在游戏设计中是不可能找到某种可以通用的按钮的，因此加入的按钮还需要引入某些素材。

标注3的地方就是这些素材的列表

标注4的部分相当于用来实现了的界面的屏幕，可以将空间直接向这里拖动来对界面或控件进行设计，也可以在这里实时观看设计出的成果。当然如果想要查看真正的界面运行效果，还是要单击“文件”菜单下方的那个三角形的按钮来执行的。

标注5是在设计时经常会需要的一些动态的效果，比如帧动画。

**（3）利用Cocos Studio实现登录APP**

本节将使用Cocos Stduio编辑器实现一个APP中的儿常见登录界面。

界面如下：



步骤1：在标记1的部分找到游戏分辨率的列表（默认为960\*640），将其中的数字修改为360\*640。将素材文件Sprite.png、textbox.png、background.png、button1.png、button2.png导入到标注为3的列表中。

步骤2：在控制面板拖动一个图片控件到屏幕上。然后右击它后在快捷菜单中选择“设置图片资源”选项，将图片background选择进来，然后调整它的尺寸。也可在标注4的面板中直接输入数据。

步骤3：可以用类似的方法在屏幕中加入除按钮外的其他元素，包括输入密码和账号的文本框和Cocos2d-x的logo图片。（可参照实现界面的布局效果）

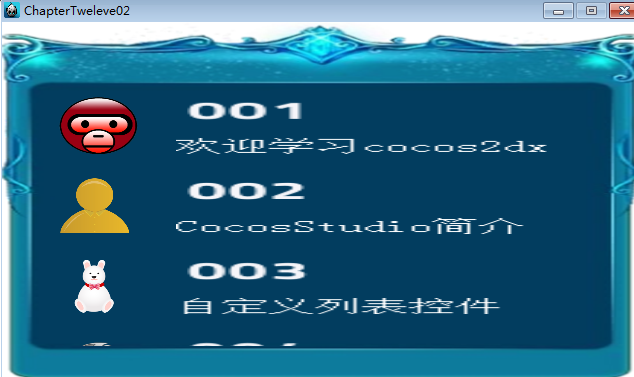
步骤4：拖动一个控件“输入框”到屏幕上去，使它刚好能够盖住屏幕上文本编辑框的素材文件。完成之后再拖动另外一个。接下来可以右击选择“编辑占位文本”命令来修改其中的文字。

步骤5：可以向其中拖进按钮控件，然后使用右键快捷菜单中“设置按钮样式”中的“正常状态”与“按下状态”命令来选择按钮在正常状态和被单击时，显示哪个图片素材的内容。

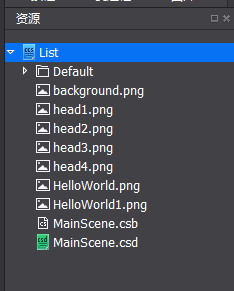
步骤6：此时单击按钮是不会有任何响应的。在VS2013中运行程序，可以测试界面的效果。

**（4）在Cocos Studio中使用自定义控件**

本节的任务就是利用Cocos Studio实现一个简单的列表控件。

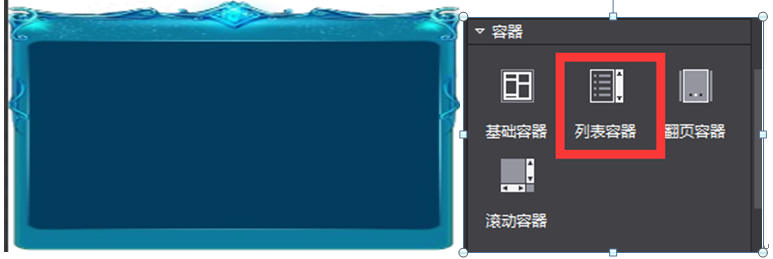


步骤1：首先创建一个新的Cocos2d-x项目，复制素材资源文件到Cocos Studio目录下并导入Cocos Studio，如下图所示。其中background.png是列表的背景图片，head1.png~head4.png是4张头像用的素材文件。

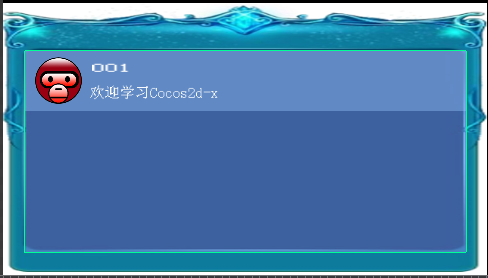


步骤2：先将背景图片background.png拖动到场景中，如下图左侧。然后可以在左侧找到容器栏的“列表容器”，如下图右侧。将列表容器拖动到背景图片中，并调整其大小使其刚好与背景图片中的边框契合。

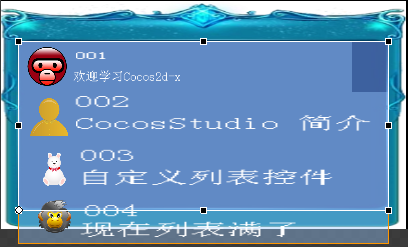
【说明】本质上来说列表容器是一个不算控件的控件，说它不是控件是因为当它没有存放内容时，它就像不存在一样，而当其加入了子控件后，又会展现出属于列表的独特性质，而且这一性质需要在列表被占满后拖动时才会体现出来。



步骤3：选中列表容器并右击，在右键弹出的快捷菜单中选择“添加子控件”|“基础容器”命令，然后修改基础容器，在其中再加入子控件，如下图所示。

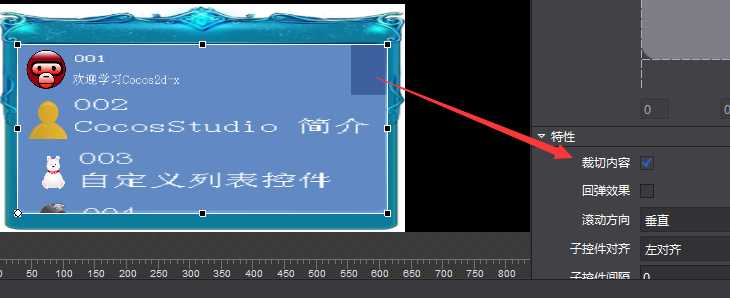


步骤4：采用相同的方法再向其中加入更多列表项。得到下图的结果。



步骤5：从界面中可以看出，列表项的内容已经超出了原本定义的列表范围。选中“列表容器”控件，在右侧的面板中找到“裁切内容”复选框，如下图所示。

【说明】可以做其他设置，如修改列表本身和其子控件的不透明度。



步骤6：此时再运行界面，看上去就好多了。将界面发布到Visual Studio中，然后运行。

**（5）对Cocos Studio制作的按钮进行响应**

上节我们学了了如何使用Cocos Studio实现某些比较复杂的界面，但是此时的界面还是只能看不能用，本节实现的界面，不仅能看而且能在Cocos2d-x中被调用。

运行程序后，界面如下图所示。当用户单击屏幕两侧的按钮时，会引起图片切换的动画。



步骤1：首先创建一个新的项目，并在其界面中加入两个按钮。并将其发布到Cocos Studio中。

步骤2：在Visual中，编辑代码。在原有代码中，引入UI界面的代码如下：

auto rootNode = CSLoader::createNode("MainScene.csb");

addChild(rootNode);

如果要获取其中的某个空间，比如左边的那个按钮，可以使用如下代码：

Button\* left = (Button\*)rootNode->getChildByName("Button\_1");

Button\_1是在Cocos Studio中为左边的按钮起的名字，可以在右侧控件属性面板的高级属性中进行查看。

步骤3：在代码中添加对按钮响应的代码，下面代码实现的效果是在场景中央显示一张图片，当单击某个按钮时，场景中央的图片会以渐变的方式移动并消失，代码范例如下图所示。

bool HelloWorld::init()

{

if ( !Layer::init() )

{

return false;

}

for (int i = 0; i < 6; i++)

{

sprite[i] = Sprite::create(String::createWithFormat("sprite%d.png", i)->getCString());

sprite[i]->setOpacity(0);

addChild(sprite[i],2);

sprite[i]->setPosition(320, 180);

}

sprite[0]->setOpacity(255);

tag = 0;

auto rootNode = CSLoader::createNode("MainScene.csb");

addChild(rootNode);

Button\* left = (Button\*)rootNode->getChildByName("Button\_1");

left->addTouchEventListener(this, toucheventselector(HelloWorld::left));

Button\* right = (Button\*)rootNode->getChildByName("Button\_2");

right->addTouchEventListener(this, toucheventselector(HelloWorld::right));

return true;

}

void HelloWorld::left(cocos2d::Ref\* pSender, TouchEventType type)

{

if (type == ui::TouchEventType::TOUCH\_EVENT\_BEGAN)

{

CCLOG("left");

sprite[(tag + 1) % 6]->setPosition(460,180);

sprite[(tag + 1) % 6]->setOpacity(0);

sprite[(tag + 1) % 6]->setScale(0.2f);

auto moveto1 = MoveTo::create(0.3, Vec2(180, 180));

auto scaleto1 = ScaleTo::create(0.3, 0.2f);

auto fadeto1 = FadeTo::create(0.3, 0);

auto spawn1 = Spawn::create(moveto1, scaleto1, fadeto1, NULL);

sprite[tag]->runAction(spawn1);

auto moveto2 = MoveTo::create(0.3, Vec2(320, 180));

auto scaleto2 = ScaleTo::create(0.3, 1);

auto fadeto2 = FadeTo::create(0.3, 255);

auto spawn2 = Spawn::create(moveto2, scaleto2, fadeto2, NULL);

sprite[(tag + 1) % 6]->runAction(spawn2);

tag=(++tag)%6;

}

}

void HelloWorld::right(cocos2d::Ref\* pSender, TouchEventType type)

{

if (type == ui::TouchEventType::TOUCH\_EVENT\_BEGAN)

{

CCLOG("left");

sprite[(tag + 7) % 6]->setPosition(180, 180);

sprite[(tag + 7) % 6]->setOpacity(0);

sprite[(tag + 7) % 6]->setScale(0.2f);

auto moveto1 = MoveTo::create(0.3, Vec2(460, 180));

auto scaleto1 = ScaleTo::create(0.3, 0.2f);

auto fadeto1 = FadeTo::create(0.3, 0);

auto spawn1 = Spawn::create(moveto1, scaleto1, fadeto1, NULL);

sprite[tag]->runAction(spawn1);

auto moveto2 = MoveTo::create(0.3, Vec2(320, 180));

auto scaleto2 = ScaleTo::create(0.3, 1);

auto fadeto2 = FadeTo::create(0.3, 255);

auto spawn2 = Spawn::create(moveto2, scaleto2, fadeto2, NULL);

sprite[(tag + 1) % 6]->runAction(spawn2);

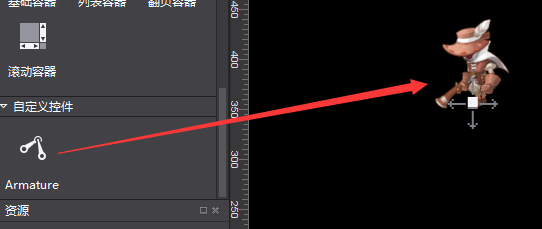
tag = (tag+7) % 6;

}

}

**（6）使用骨骼编辑器制作动画（自学）**

在Cocos studio中有一个看上去非常高端的控件，如下图所示。将它拖动到场景中会出现一个人物，右击，并在弹出的菜单中选择“播放”命令，它就会在屏幕上走动。这个控件需要一个额外的骨骼动画编辑器生成



安装骨骼动画编辑器

什么是骨骼编辑器：先从之前讲过的帧动画说起，帧动画的本质是连续地切换不同的图片资源，骨骼动画与帧动画不同，因为它确实在动，是由设计人员设计的骨架（即骨骼）带着绑定在骨架上的素材图片一起运动。