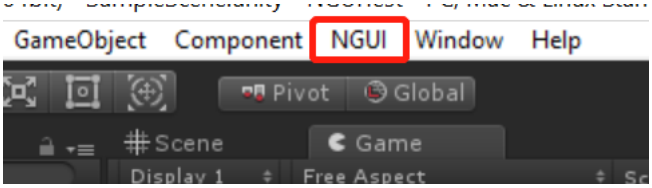
Unity 3D游戏编程自学#7——NGUI入门

原创 Zerg_Wang 于 2019-02-24 20:12:17 发布 231 收藏 编辑 版权

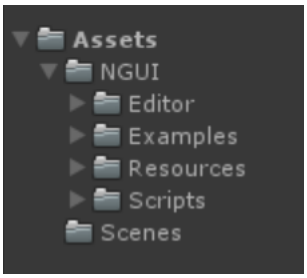
分类专栏： Game Programing

1. NGUI简介

NGUI: Next-Gen UI kit（以下简称NGUI）是一个第三方的Unity开发包（我原来以为是Unity自带的），下载下来后是一个unitypackage的文件，导入后会发现工具栏多了一个NGUI的菜单：



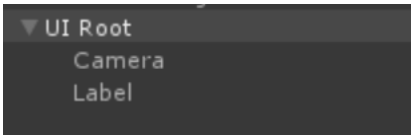
在Assets文件夹中也多了一个NGUI的文件夹，其中Editor是编辑器拓展， Examples是示例工程（打开Examples下的Scenes中的场景，即可体验），Resources是资源文件，Scripts是脚本组件（重点）。



2. NGUI Label

1. 界面简介

NGUI→Create→Label，用于显示文字UI，之后可以看到NGUI文件的层次结构：

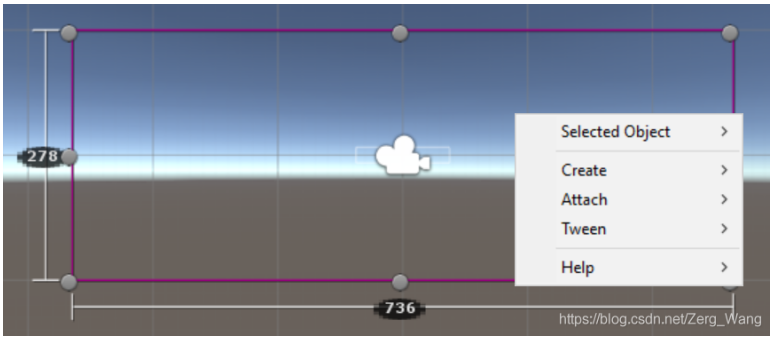


UI Root是所有NGUI元素的父物体，旗下的Camera为专门用于渲染NGUI的摄像机，其他的Label等等的就为NGUI的元素了。

（若要单独创建UI Root，则NGUI→Create→2D UI）

此时在Scene面板中可以看到一个紫色矩形，这个矩形就是UI Root，NGUI物体会在这个矩形内显示。

切换2D显示模式，选中UI Root，会发现矩形高亮并显示出其长宽信息，这时在矩形内部右键就可以对该UI界面进行进一步操作。



如果摄像机图标过大妨碍观察，可使用Scene面板右上角Gizmos下的3D Icons将其调小。

接下来NGUI→Open→Prefab Toolbar可打开预制体工具栏，里面都是事先制作好的一些UI元素，可直接拿来使用。（单击选中拖拽到Scene面板的矩形中）

2. 文字编辑

选中Label，右侧的UI Label(Script)栏中可以进行文字编辑，第一项可以选择NGUI或者Unity，表示显示的字体是使用Unity自带的还是NGUI自带的（NGUI自带的是字体图集，即把字体做成图片用于显示，在Unity中以预制体的形式保存，Unity自带的是ttf格式的字体），点击Font即可选择字体。

Font Size控制文字大小（后面的那个选项用于调整显示效果，例如粗体、斜体），与之相关的是下面Widget里面的Size，这个调整的是文字显示框的大小，如果文字大小超过了显示框，则在Overflow中可以选择：

Shrink Content：当文字大小超过显示框，则文字大小不变，Font Size的调整无效。

Clamp Content：文字正常放大，超出显示框的文字不予显示。

Resize Freely：文字正常放大，显示框大小与文字大小同步变化以显示所有文字。

Resize Height：同上，但显示框仅高度自由变化，宽度不变。

Text栏用于输入要显示的文字（可回车后输入多行），Spacing调整文字水平、垂直间距（若x为负数，则文字左右倒序，若y为负数，则上下交换）

Modifier用于控制英文的显示（正常、全大写、全小写），Alignment调整对齐方式。

Gradient和Color Tint都是调整文字颜色的，两者只能选其一，前者可使用渐变色。

Effect为特效，如颜色描边、阴影等。

在游戏中有些显示文字可能会变化（例如玩家金钱、等级，多为数字），文字长度也可能变化，因此一般要把Overflow设置为Resize Freely，此外也要在Widget中设置文字中心点（Pivot，显示内容变多或变少时显示框相对变化的地方），默认是显示框的水平中心和垂直中心，假如调整为水平左侧，文字变多的话会向右延伸：（第三个为调整后的）

在脚本中也可以控制以上这些内容：

```
private UILabel test;
void Start()
{
    test = GameObject.Find("Label").GetComponent<UILabel>();
    test.text = "12345";
    test.color = Color.red;
}
```


若要输入多行，\n换行即可。

其他的就不再演示了。

3. NGUI Sprite

1. Atlas图集编辑工具

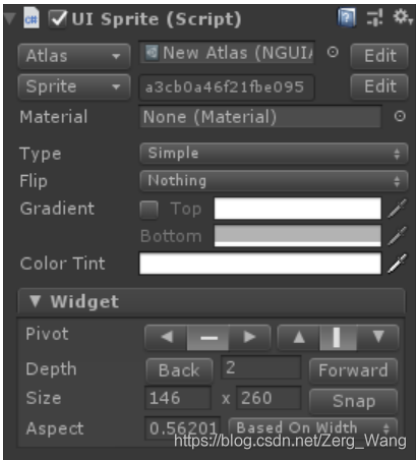
对于游戏中要用到的图片，可用NGUI自带的Atlas Maker编辑成图集。

NGUI→Open→Atlas Maker，选择好要编辑的图片，点击Create即可，Atlas Maker会在指定文件夹内生成三个文件：png是所有图片拼接成的一张大图，mat为以png制作的材质球，prefab为以材质球制作的预制体。之后若要使用，则会直接调用预制体。

若要添加新的图片或要删除已经制作成图集的文件，则打开Atlas Maker直接进行增删操作即可。

2.Sprite简介

用于显示图片的UI，NGUI→Create→Sprite创建，在UI Sprite(Script)面板中可以选择要显示的图集（Atlas）中的某张图片（Sprite），Widget中的Size可以调整图片的大小，点击Snap会按图片原大小进行显示。



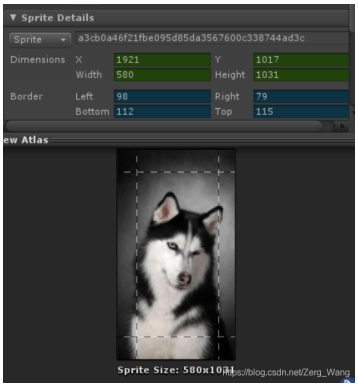
Size后面的Aspect是图片长宽比例，后面的框有三个选项，Free表示可自由调整图片大小，后面两个都是等比例缩放，Based on Width表示只能通过调节宽度进行缩放（长度会跟着变，但不能直接调节长度），后面的Based on Height类似。

Flip为图片是否翻转，Gradient和Color Tint与Label类似（这个颜色是在原图基础上叠加的颜色，若不想使用此效果，设成白色即可）。

Type选择图片的展示方式：

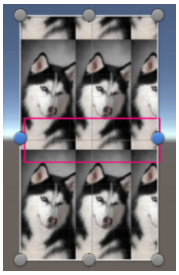
Simple：默认展示方式，无特殊效果。

Sliced：九宫模式，选中此模式后点击Sprite后面的Edit，在新界面中设置Border，可把图片分成九部分：



之后再进行缩放操作，就只有九宫的中间部分会进行，其余八部分则不变。这种显示模式非常适合一些与文字组合的图片UI（可以随着文字的增减而改变大小）

Tiled：平铺模式，图片反复拼接，直到铺满Size区域。（在此模式和Sliced下Snap按钮失效）



Flip：进度模式，只有在此模式下Fill Dir和Fill Amount才可用，Fill Dir为填充方向，Fill Amount为填充量，1表示图片完整显示，0表示图片不显示，可收手动调节，具体效果不好描述，自己搞搞就明白了。

3. 代码控制

目标：创建一个空物体，使其成为UI Root的子物体并为其添加UI Sprite组件，显示指定图片。（脚本挂载在UI Root上）

```
1  private Transform rootTransform;
2  private Transform imageTransform;
3  void Start ()
4  {
5      rootTransform = gameObject.GetComponent<Transform>();
6      GameObject showImage = new GameObject("ImageOne");       //新建一个游戏物体用于显示图片
7      imageTransform = showImage.GetComponent<Transform>();    //得到该游戏物体的Transform组件
```

```
    imageTransform.SetParent(rootTransform);                //设置父子关系
    imageTransform.localScale = Vector3.one;                //把新游戏物体的Scale设为1,Vector3.one为(1,1,1)
    UISprite imageSprite = showImage.AddComponent<UISprite>();
    UIAtlas imageAtlas = Resources.Load<UIAtlas>("MyAtlas");//通过Resources.Load指定图集
    imageSprite.atlas = imageAtlas;                         //令imageSprite获取图集
    imageSprite.spriteName = "123";                         //指定要显示的图片（通过图片名）
}
```

这里需要注意的是，为了使用Resources.Load，图集预制体要放在Assets下的Resources文件夹中，然后Load后面尖括号内写明加载的资源类型，圆括号内写明资源名（无需后缀名）即可，如果Resources里面还有n层文件夹，则资源名处要写明路径（相对路径）。

这段代码我其实调了很久，一直遇到的问题是无法加载图集，这是因为我使用的NGUI版本为v2018.3.0e，Atlas Maker只能制作.asset的图集，不能制作.prefab的图集，这样一来Resources.Load就无法识别，最后换成了v3.12.1版本才解决了问题。

4. NGUI Button

1. 创建Button

在Label或Sprite上附加的组件，添加方法：在Scene面板中对要添加的Label或Sprite右键Attach→Box Collider，然后再对该物体右键Attach→Button Script。（下面的例子是在Sprite上完成的）

在Box Collider中可以看到系统自动地将可点击区域与图片大小设置为一样。

在UI Botton中有Colors和Sprites两个主要选项，其实都是用于编辑按钮的不同状态的，Normal为按钮的正常状态，Hover为鼠标移动到按钮上的状态，Pressed为按下的状态（左键、右键、中键均会触发，该状态会持续到鼠标抬起或移开），Disabled为按钮不可用状态。Colors是编辑按钮在不同状态下的颜色（同样是叠加的颜色），Sprites为不同状态下显示不同图片。不过有个小细节：当Button处在不同状态时，会发现Normal这一栏的图片会变成其他状态下的图片，也就是说button不同状态下图片的切换时通过更改Normal来完成的。

此外，可通过右键Attach→Play Sound Script（Add Component添加亦可 ），为按钮添加点击音效。若要按钮在不同状态下有不同音效，多添加几个即可。

3. 代码控制

方法一：（脚本可挂载到任意物体上）

```
public void ButtonDown()
{
    Debug.Log("1");
}
```

然后找到Button的Inspector面板中UI Button下的On Click，从挂载了该脚本的物体的Inspector面板中把脚本拖拽至On Click（不能直接从Assets文件中拖过去），然后选择对应的方法（这里是ButtonDown）即可。

方法二：（脚本只能挂载在Button上）

```
void OnClick()
{
    Debug.Log("2");
}
```

OnClick是NGUI自带的方法，直接拿来用即可。

5. NGUI总结

在实际开发过程中，制作好的UI会单独抽出做成预制体，不会和Scene粘连。一般把要做成预制体的UI先以父物体Plane包含：

然后拖拽至Resources文件夹中做成预制体。

在需要使用UI的场景中，再将其Resources.Load：

```
private GameObject prefabP;
void Start ()
{
    prefabP = Resources.Load<GameObject>("Panel");
    NGUITools.AddChild(gameObject,prefabP);
}
```

后面那句是NGUI自带的API，在实例化Panel的同时将其设为UI Root的子物体。（该脚本挂载在UI Root下）

AddChild前一个参数为父物体，后一个为子物体。

本文部分内容来自擅码网（http://www.mkcode.net）Unity 3D课程，经本人学习、整理得来，若有错漏，欢迎指正！