NGUI三大基本组件

UIRoot UICamera UIPanel

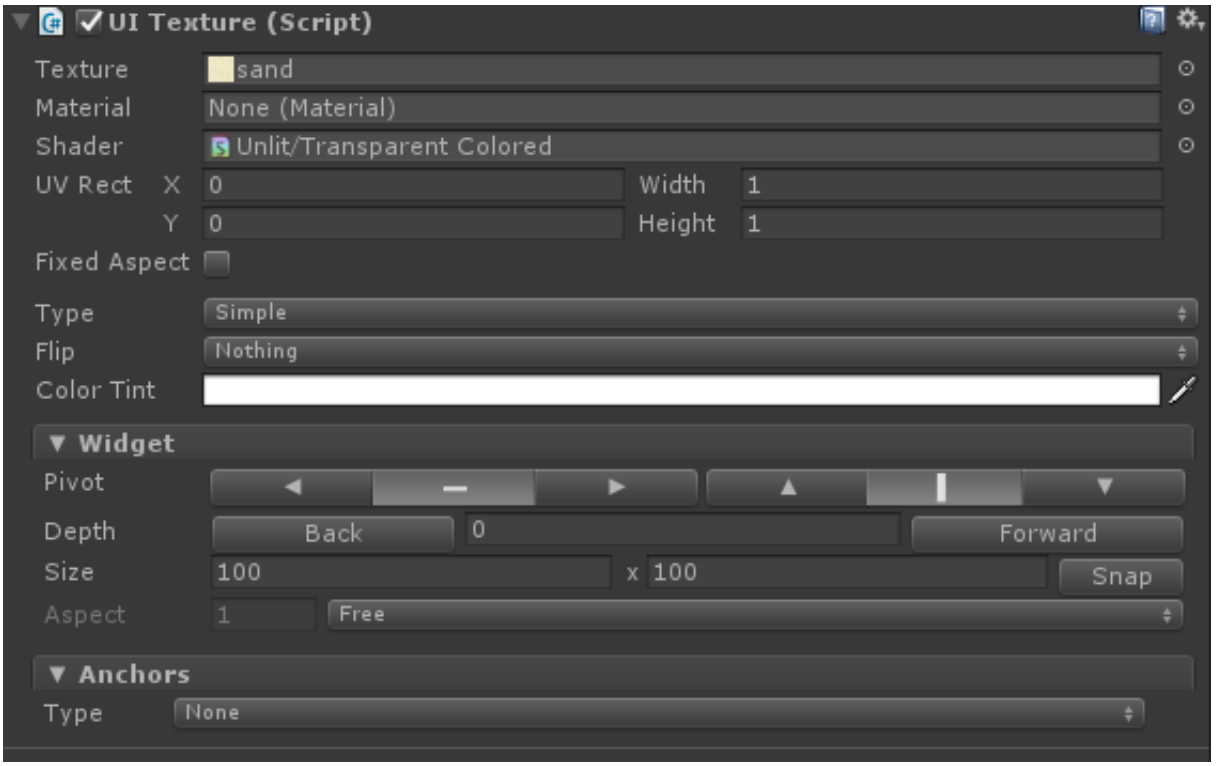
UIRoot 制作UI时 屏幕分辨率适配的问题 控制UI的缩放模式

UICamera 负责监听UI模块的事件 比如点击事件 拖拽事件 输入事件等

UIPanel UI界面管理组件 控制UI面板的显示效果 比如 透明度和深度

UITexture UI纹理

是最基础的NGUI渲染组件 UIWidget



UIPanel UI控制面板

用来收集和管理它所有的子节点（Widget组件）

所有需要显示的UI物体 都必须放在Panel的子节点下

没有Panel所有的UI控制都不会被渲染 可以把Panel当做是Renderer

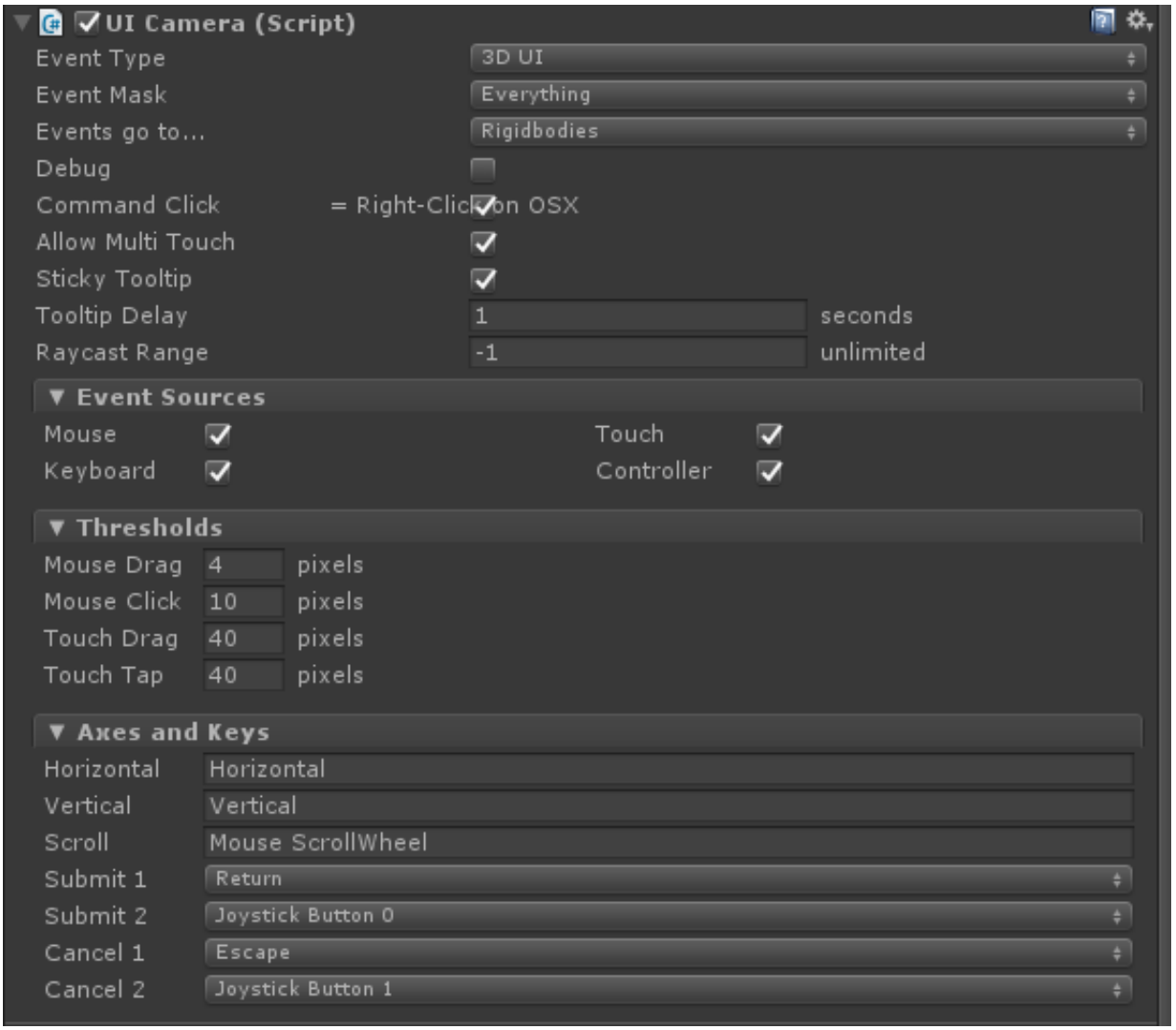
UIPanel可以有多个 在制作游戏时 以你为场景中会出现多个UI窗口 最好的处理方式就是添加多个Panel

主要的两个值：Alpha(透明度）Depth（层级 深度 渲染顺序）

如果多个UI界面 需要控制显示层级 添加多个Panel

调整Panel的Depth Depth高的会覆盖Depth低的

UICamera UI摄像机



EventType 一般选择2DUI 事件的监听顺序和Depth相关

UICamera如果是绑定在MainCamera上 设置成3DUI 事件的监听顺序和距离摄像机的值有关

EventMask 决定哪些Layer的物体接收事件

Debug 用来显示鼠标指向的物体是什么

Allow Multi Touch 是否支持多点触控 不勾选即使是多点触控也会当做单点触控来使用

RaycastRange 射线碰撞的检测范围 -1 不限制距离

NGUI主要的事件函数

Collider(Trigger)必备组件

OnHover（isOver）

当鼠标悬停（离开）在一个collider上时调用

OnPress(isDown)

当鼠标在一个Collider上按下

OnSelect（selected）

鼠标 点击和抬起都在同一个Collider上

***OnClick()**

发送时机和OnSelect时机一致 但是要求鼠标不能移动太多

OnDoubleClick()

发送时机：当在四分之一秒内Click两次的时候调用

OnDragStart()

开始拖拽

OnDrag()

拖拽中

OnDragOver()

其他的object拖拽到我的上面

OnDragOut()

其他的object从我的上面拖拽出

OnDragEnd()

当拖拽事件结束

OnInput(text)

输入的事件函数

OnToolTip(show)

鼠标悬停在一个Collider上一段时间没有移动

OnScroll(float delta)

鼠标滚轮滚动

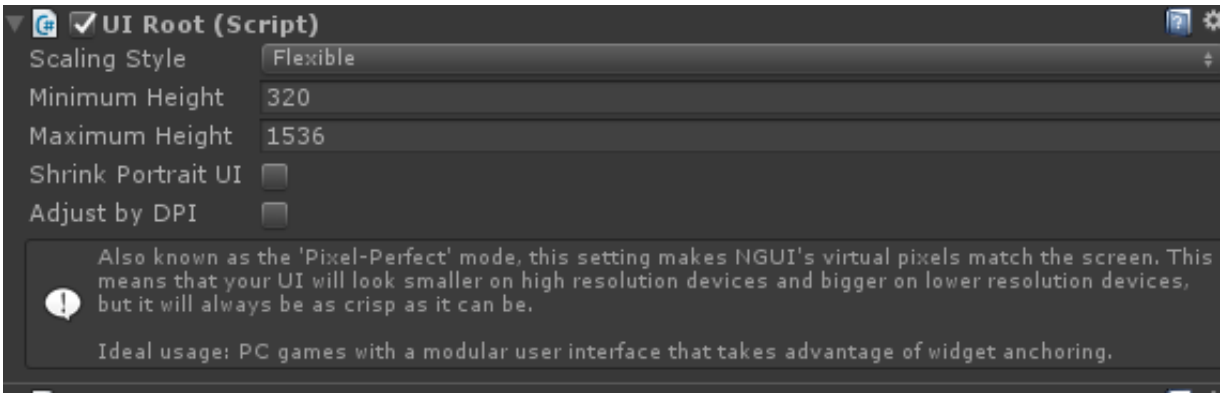
OnKey(keyCode key)

键盘或者输入控制器被使用的时候

UIRoot

UIRoot 一般用于界面的根节点

目的解决不同分辨率下导致UI显示问题 实现UI适配



Scaling Style 缩放模式

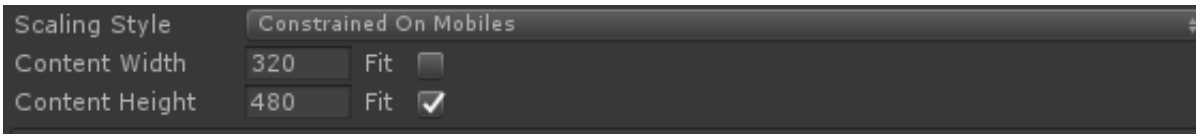
Fiexible 自由缩放模式 适用于PC

Constrained 限制缩放模式 适用于移动端

Constrained On Mobiles 二者的综合 自动判断平台设置不同缩放模式

移动端适配原则

有基于宽度和基于高度两种 模式 一般选择基于高度Fit Height



在ContentWidth和ContentHeight上填写 设计分辨率320*480

当设备分辨率为640*960时 完美显示 基于高度放大了2倍

当设备分辨率为600*960时 显示效果：高度显示正常 宽度会被裁减

原则上 在做屏幕适配时 不可以出现UI显示不完整的情况

所以此时需要调整摄像机的Size 保障UI完全显示

当调整后效果： 上下会留出黑边 但是界面显示完整

所以在开发过程中 做UI适配 除了让Root帮助我们事先界面缩放以外

还需要根据当前的设备宽高实时调整摄像机的Size

绑定在UICamera上

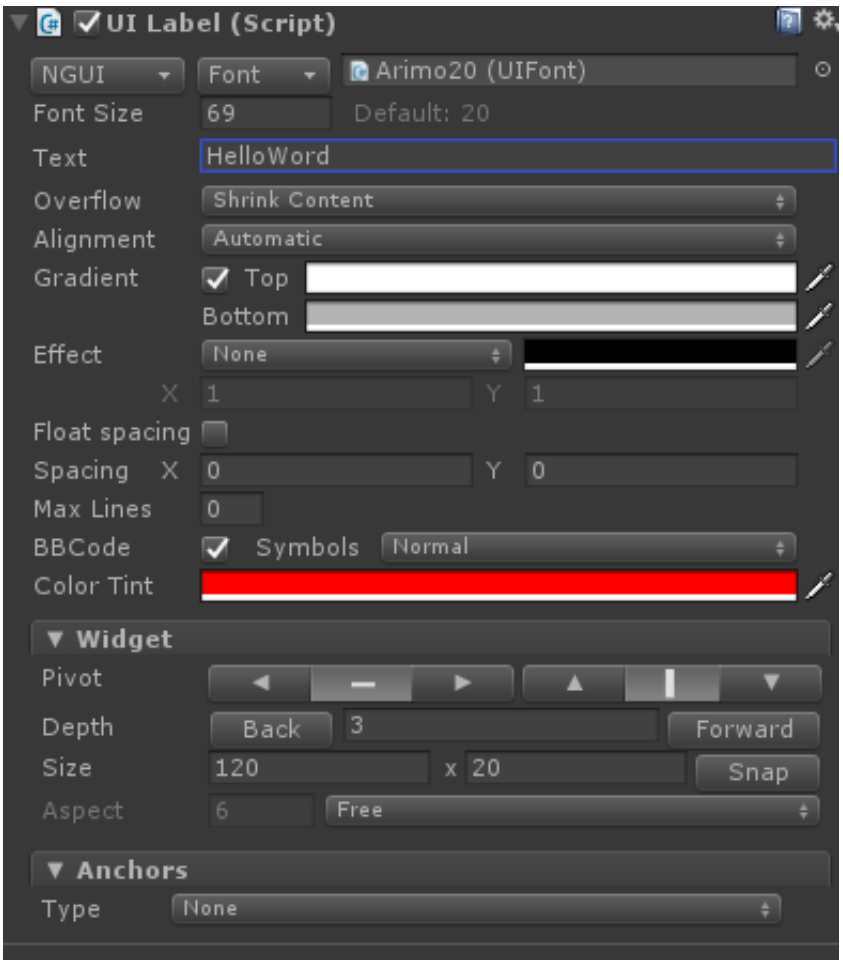
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraControl : MonoBehaviour {
```

```
6
7       //设计分辨率的标准宽高
8       private float standard_Width = 320;
9       private float standard_Height = 480;
10
11      //运行时 设备的宽高
12      private float device_width;
13      private float device_Height;
14
15
16      //UI摄像机
17      private Camera camera;
18
19      void Start () {
20          camera = GetComponent<Camera>();
21          device_width = Screen.width;
22          device_Height = Screen.height;
23          SetCameraSize();
24      }
25
26      void SetCameraSize()
27      {
28          float adjustor = 0f;
29          float standard_aspect = standard_Width /
    standard_Height;//320/480=0.666
30          float device_aspect = device_width / device_Height;
31
32          //设计标准比例  320/480 =0.6666
33          //当设备分辨率是640*960时  比例也是0.6666 不需要调整Size
34
35          //当设备分辨率是660*960时 比例是0.6875 大于设计宽高比 此时 两边留
    出黑边 但是界面正常显示 不需要调整
36
37          //当设备分辨率是 600*960时 比例是0.625 小于设计宽高比 此时 UI显示
    不完整需要调整Size
38
39          if (device_aspect < standard_aspect)
40          {
41              adjustor = standard_aspect / device_aspect;
42              camera.orthographicSize = adjustor;
43          }
44      }
```

```
45 }
46
```

UILabel 文本标签



FontSize 字体大小

Text 文本内容

overFlow 填充内容选项

ShrinkContent 以内容为基准填充 会自动缩小以便适应矩形区域

ClampContent 以字体大小为基准 进行填充 会自动截断

ResizeFreely label大小由文本长度来控制 不受矩形的限制

ResizeHeight 必要时增加label的高度 宽度保持不变

AlignMent：字体对齐方式

Gradient:字体渐变

Effect 字体效果

Shadow 阴影

outline 外边框 阴影和外边框都会增加多倍几何

spacing 字体间距

MaxLines 字体行数限制 0是不限制

BBCode: 是否需要处理颜色标签和表情符号

:) :D <_< >_< x_x -_- o.o

(A)(B)(X)(Y)

NGUI的中文支持

NGUI的字体分为两种：NGUI Unity

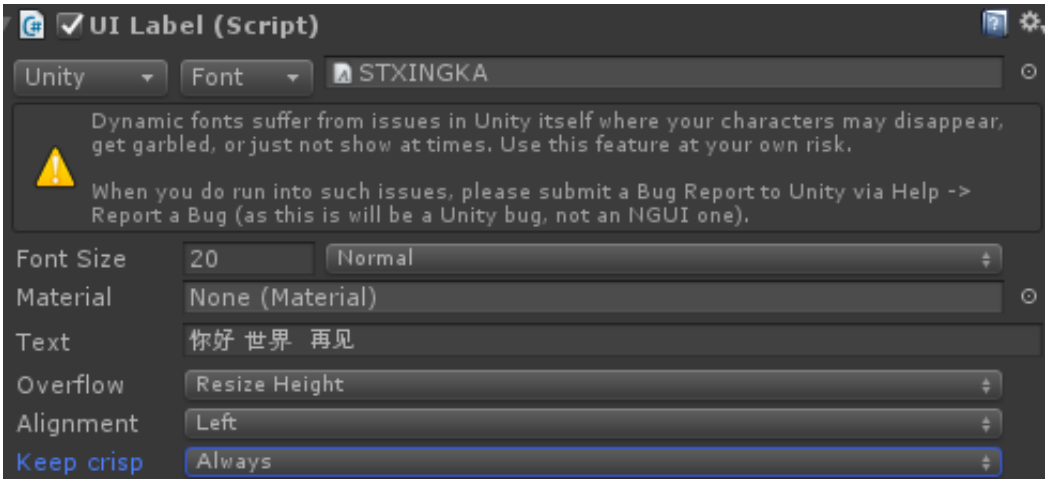
NGUI一般是指美术字体 想支持中文 比如把美术字作为图片添加到材质中

Unity一般是指系统字体 可以直接使用中文

Label支持中文的方式

1 在Windows中查找支持中文的系统字体

2 UIlabel控制中的NGUI该为Unity并赋值



Label文本转义：

文字加粗：[b]bold[/b]

文字斜体格式：[i]text[/i]

文字加下划线：[u]text[/u]

直线穿过文字：[s]text[/s]

左下角或右下角显示 [sub] [sup]

在同一段文字中设置不同的颜色

0,0,0-255,255,255 0,0,0-ff,ff,ff

[ff0000]text[-][00ff00]text[-][0000ff]text[-]

Label点击事件的问题

NGUI中接收点击事件的条件：

1 添加触发器

2 设置Layer 和UICamera保持一致

3 重写事件函数 OnClick

点击Label获取超链接 打开网页

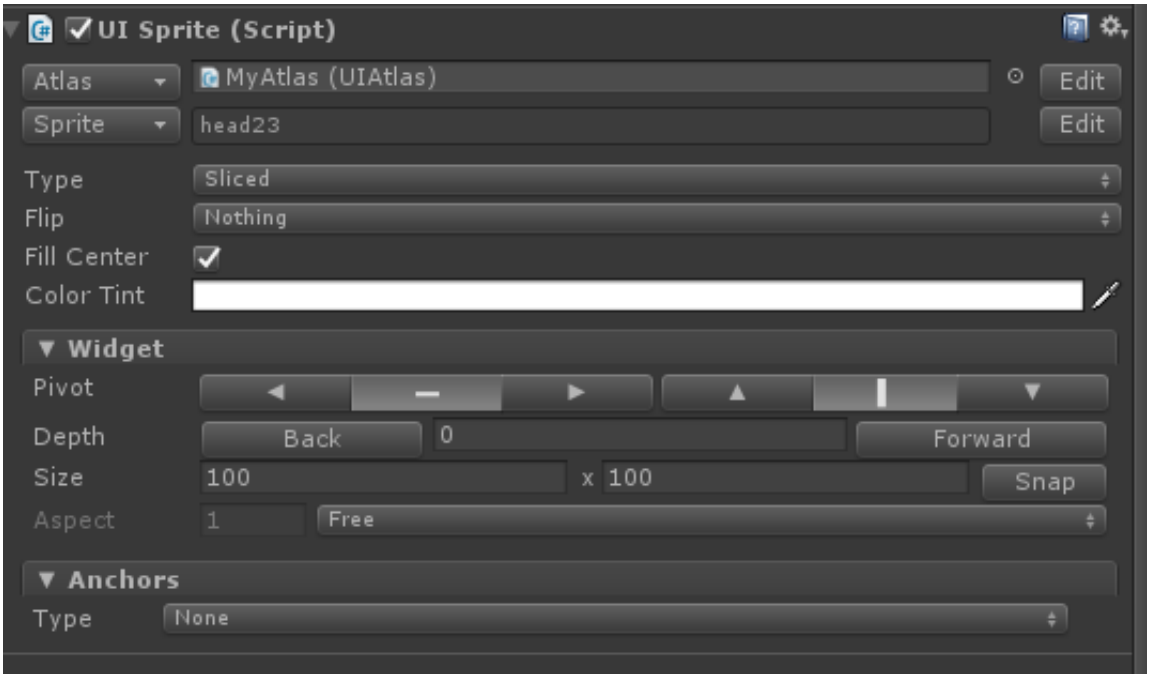
Label的text内容 [url=http://www.baidu.com/][u]进入官网[/u][/url]

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LabelText : MonoBehaviour {
    UILabel label;
    void Start()
    {
        label = GetComponent<UILabel>();
    }
    void OnClick()
    {
        label.color = Color.green;
        Invoke("Fn", 0.2f);
        //根据点击位置 获取超链接
        string url=label.GetUrlAtPosition(UICamera.lastHit.point);
```

```
17
18          //打开超链接
19          Application.OpenURL(url);
20      }
21      void Fn()
22      {
23          label.color = Color.red;
24      }
25  }
26
```

UISprite 精灵



如果使用一个精灵 必须先把精灵添加到 build到 Atlas中

NGUI的Atlas --图集 默认就是一张宽高为2的N次幂的纹理

选中多个图片 ->OpenAtlasMaker->Create

Atlas生成时 创建一张纹理 一个材质 一个预设

在开发中尽量使用UISprite 少使用UITexture

使用Altas可以对DrawCall进行优化

DrawCall

渲染工作是CPU和GPU并行工作

DrawCall 就是CPU调用底层图形绘制的接口(OpenGL或者Direct3D) 通知GPU进行渲染的操作

CPU在通知GPU渲染以前 会执行很多的准备工作(设置颜色，绘图方式，顶点坐标等信息)

检测渲染状态 提高渲染数据 提交渲染状态

当DrawCall过多时 CPU会造成很多的额开销 花费大量的运算在DrawCall的准备工作上

所以在开发中 DrawCall的次数是决定性能的一个重要指标

合并纹理 （ 图集 ）

把很多小的DrawCall合并到一个大的DrawCall上 这就是Atlas的优化原理