UGUI界面开发技术

一：UGUI概述

二：UGUI基础控件

三：UGUI高级控件

四：Anchor锚点与屏幕自适应系统

五：综合案例分析

一：UGUI概述

UGUI是Unity4.6之后，经过多重测试，推出全新的UI系统，更灵活，快捷，易用的可视化游戏UI开发工具。

由于之前传统的UI系统存在很多诟病，因此出现了很多UI插件，其中比较出名的是NGUI 、Easy GUI，当然也有其他的UI插件。

UGUI在吸收第三方插件的优秀编程思想的基础上，整合Unity内部强大的技术体系，使得UGUI成为非常优秀的UI开发技术与标准。

（1）与Unity引擎无缝紧密结合

（2）更加强大与易用的屏幕自适应能力

（3）更加简单的深度处理机制

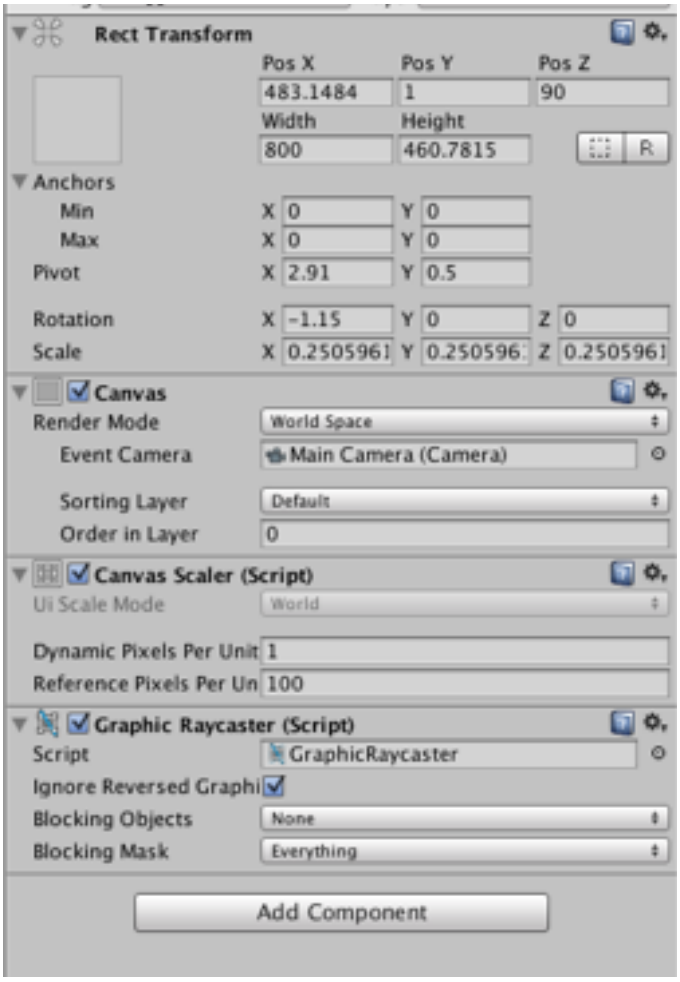
（4）完全自动化的图集打包功能

Editor ->Project Settings ->Editor Settings里面有一个sprite packer的模式。Disabled 表示不启用，Enabled For Builds 表示只有打包的时候才会启用它，Always Enabled 表示永远启用它。这里的启用它就表示是否将小图自动打成图集。

（5）全新强大的布局系统，简单易用的UI控件，强大与易用的事件处理系统

二：UGUI基础控件

UGUI控件在系统菜单中具有11个控件，通过自由组合还可以创建出功能更强大的复合控件。下面我们介绍常见的一些控件。



2.1 Canavs画布

Canvas画布是UGUI系统最基础的容器类控件，所有的UI控件必须位于Canvas画布控件之内，即必须是Canvas容器的子控件。

我们单击 UI->Canvas来创建画布。添加完成之后，可以看到在当前的Hierarchy视图中，添加了Canvas和EventSystem。

Canvas控件具备4个组件，RectTransform,Canvas,CanvasScaler,GraphicRaycaster。下面分别介绍每一个组件的功能。

（1）RectTransform：可以将其理解为矩形变幻，与普通对象的Transform相对应，主要针对UGUI界面专门提供的界面参数组件，包括UI界面的大小，旋转，中心点，相对中心点偏移量，设置锚点等参数。

（2）Canvas组件提供了画布渲染，像素完善，画布排序等功能。其中渲染模式是一个重要的参数。我们分别对其进行讲解。

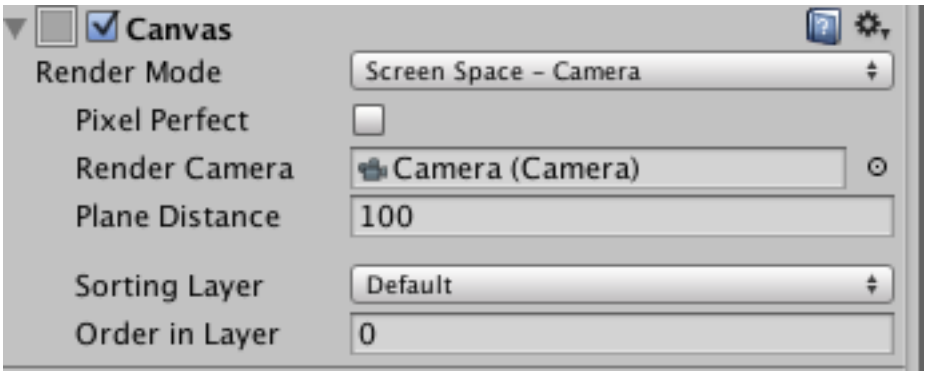
Render Modes 渲染模式

- Screen Space - Overlay

此模式不需要UI摄像机，UI将永远出现在所有摄像机的最前面，类似于给摄像机镜头前添加了一层透明膜。

- Screen Space - Camera

此模式需要专门为UI指定一个Camera。此种模式是最常用的模式，它允许UI界面前可以显示其他游戏对象，如为了增加特效而添加的粒子系统。摄像机与UI之间的距离通过Plane Distance来进行设置。



- World Space

此种模式下，将UI当3D对象来对待。我们移动摄像机的时候，发现UI并没有随着摄像机一起移动。这个是完全的3D的UI

- Pixel Prefect 像素完美，点对点（让我们的像素和平面像素对应）
- Sort Order 排序次序 （一般情况下无需修改）

（3）Canvas Scaler组件

Canvas Scaler 组件表现为画布的大小，其中UIScaleMode有三种模式。分别是下面的三种。

- Constant Pixel Size 固定像素尺寸

  scale factor 比例系数

  Reference Pixels Per Unit 参考像素单位

- Scale With Screen Size 按照屏幕大小（自适应）

  Reference Resolution 相对分辨率

  Screen Match Mode 屏幕匹配模式

  Match Width Or Height 匹配宽度或高度

  Expand 扩展延伸，扩容

  Shrink 收缩，收容

  Reference Pixels Per Unit 参考像素单位

- Constant Physical Size固定物理尺寸

  Physical UInit ，物理单位 Centimeters厘米 、Millimeters毫米、Inches 英寸 、Points 点

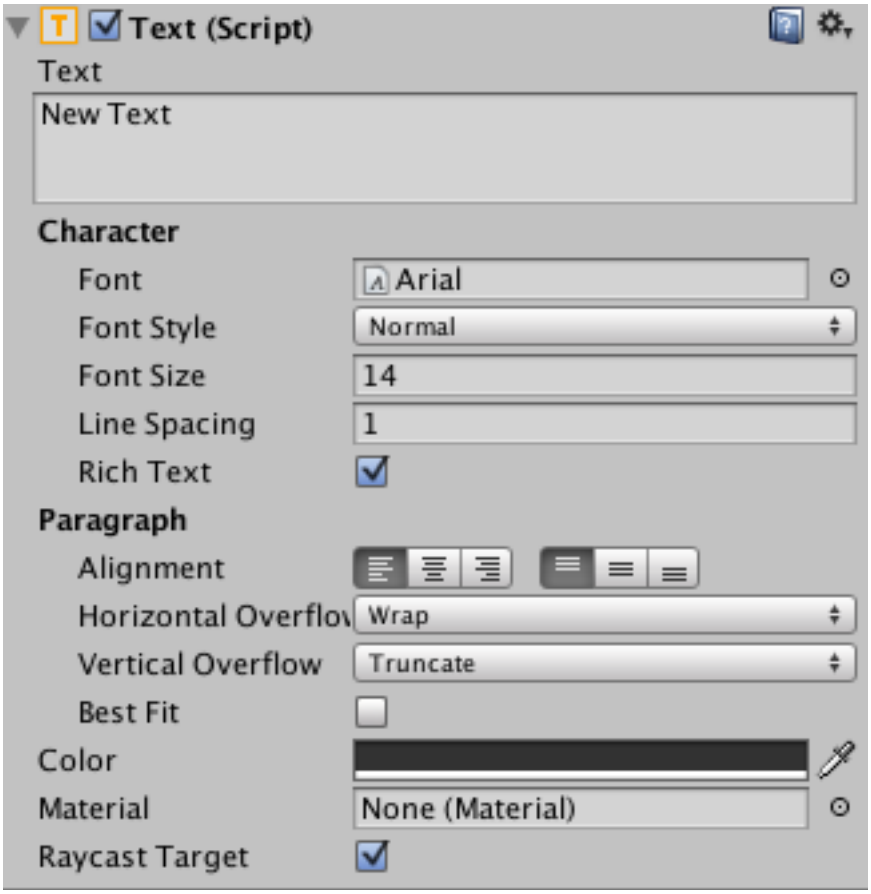
  Fallback Screen DPI 屏幕DPI回落 （DPI 分辨率表示每英寸点数）

  Default Sprite DPI 默认的精灵DPI

  Reference Pixels Per Unit 参考像素单位

（4）EventSystem组件

EventSystem是控制UI界面总体的事件管理器，分别表示UI事件系统，输入模块系统，触摸输入系统。



2.2 Text控件

Text控件属于最常用的控件之一，参数如上图所示，具体属性会在下面进行介绍。创建一个文本，我们直接进行属性的演示与讲解。

- Font 字体（字体格式.ttf）
- Font Style 字体样式
- Font Size字体大小
- Line Spacing 行距 （多行）
- RichText富文本支持
- Paragraph 段落
- Alignment对齐方式
- Horizontal Overflow 水平溢出

  Wrap 截断 Overflow 溢出
- Vertical Overflow 垂直溢出

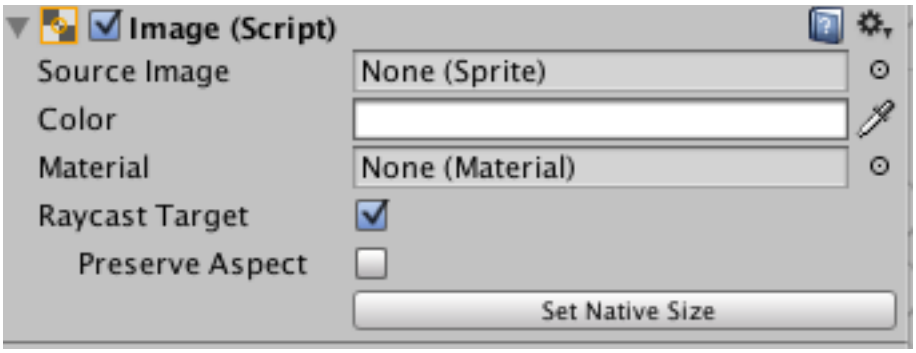
Truncate 截断　Overflow 溢出

- Best Fit 字的自适应

- Color 字的颜色（可根据需求更改）
- Material 材质（一般不需要）

2.3 Image控件

Image属于基本的控件，界面的背景，Button的背景以及很多都可以使用Image。



- Source Image　源图像　（可以给一个精灵）

精灵制作：选中要制作精灵的图片，对于图片的格式，选择“Sprite（2D and UI）”点击“Apply”即可（其他参数可不理会）。

如果觉得图片有点小，可以更改Max Size为2048，Format为trueColor ，这样就不会进行压缩了。

- Color：贴图颜色
- Material ：材质
- Image Type：图片类型，其中图片类型分为四种，Simple,Sliced,Tiled,Filled

（1）Simple普通类型

Preserve Aspect：图像的高度和宽度保持比例还是重新调整（如果想缩放但不想拉伸，请点选此项）

SetNative Size：设置原始大小 （如果想显示原始大小，请点选此项，也可以更改RectTransform的Width和Height的值）

（2） Sliced 切片精灵

切片具有良好的UI显示性能，因为当图像缩放后其边界保持不变，这种特性允许你显示图像的轮廓。不用担心放大与缩小的同时轮廓变化。如果你只想要没有中心区域的边框，则可以禁用FillCenter选项。

当我们选择这个选项的时候，可能会有一个警告（This Image doesn’t have a border），这个时候，我们需要将图片进行处理。如果没有警告，说明图片是已经处理好了，可以忽略。

制作切片精灵需要以下步骤：

- 单击需要做切片的图片，在属性窗口单击SpriteEditor,会出现下图界面。
- 在弹出的窗口中，标出边框，单击Apply按钮。
- 再次进行Slice设置的时候，警告消失。

（3）Tiled 平铺

图像保持其原始大小，重复多次填补空白。

（4）Filed填充

填充类型可以制作常见的贴图特效，也可以制作技能冷却

- Fill Method填充方式

  Horizontal 水平的

  Vertical 垂直的

  Radial 径向、射线（半径）

Radial 90　半径90度

Radial 180　半径180度

Radial 360　半径360度

- Fill Origin　填补起源

Buttom 下方

Right 右方

Top上方

Left 左方

- Fill Amount　填补数量
- Clock Wise　顺时针方向

2.4 Button控件

Button控件是一个简单的复合控件，其按钮上的文字是由内部的子控件Text负责展示，按钮的外观由组件Image负责，按钮的行为与事件由Button组件负责。我们把重点放在Button的组件介绍上。

- Interactable 是否启用（具备交互性）
- Transition 过渡方式

常见的过度方式有三种，ColorTint，SpriteSwap，Animation。下面分别介绍一下每一种的常见属性。

（1）Color Tint 颜色色彩，色彩化（默认状态最常用）

Target Graphic　目标图像
Normal Color　正常的颜色
Highlighted Color 鼠标经过时的颜色
Pressed Color　鼠标点击时的颜色
Disabled Color　禁用时的颜色
Color Multiplier　颜色倍数
Fade Duration　变化过程时间（几秒后完成颜色过渡）

（2）Sprite Swap 精灵交换 （需要使用相同功能不同状态的贴图）

Target Graphic 目标图形
Highlighted Sprite 鼠标经过时的精灵
Pressed Sprite 鼠标点击时的精灵
Disabled Sprite 禁用时的精灵

（3）Animation 动画（最复杂，效果最绚丽）

Normal Trigger 正常触发
Highlighted Trigger 鼠标经过时的触发
Pressed Trigger 鼠标点击时的触发
Disabled Trigger 禁用时的触发
Auto Generate Animation 自动生成动画

（4）Navigation 导航

None 无
Horizontal 水平
Vertical 垂直
Automatic 自动
Explicit 明确的，清楚的

在Button组件的下方有一个On Click()选项，这就是Button控件处理事件的重要机制。将挂载脚本的对象拖入，并且设置对应的函数就可以触发Button对应的事件。

在实际的项目开发之中，我们有时候需要进行所谓的动态事件处理机制，也就是说我们需要动态的创建一个Button控件，然后给Button控件添加事件处理机制，如下代码所示：

```
public class SimpleDynamic : MonoBehaviour {
    public GameObject goParent;
    // Use this for initialization
    void Start () {
        GameObject GoNewObject = new GameObject ("Button");
        GoNewObject.transform.SetParent
(goParent.transform,false);

        Image image = GoNewObject.AddComponent<Image> ();
        Button btn = GoNewObject.AddComponent<Button> ();
```

```
            image.overrideSprite = Resources.Load
("coinn2",typeof(Sprite)) as Sprite;
            btn.onClick.AddListener (ProcessSomething);

      }
      void ProcessSomething(){
            print ("Button is Clicked");
      }
      // Update is called once per frame
      void Update () {

      }
}
```

2.5 Input Filed

Character Limit 输入字符的数量

Content Type　输入内容的类型

Standard　标准，规范

Auto Corrected 自动校正

Integer Number 整数

Decimal Number 小数，十进制数，十进制小数

Alphanumeric　字母

Name 名字，名称

Email Address 电子邮箱

Pass Word 密码

Pin　接口类型，个人识别号码

Custom　自定义

（1）创建Image，拖入事先准备好的背景图片，适当的调节大小

（2）创建text，如入事先准备好的字体，调节字的大小、格式

（3）创建InputField，修改Content Type为字母，调节字的格式，大小

（4）创建InputField，修改Content Type为密码，调节字的格式，大小

（5）创建Button，修改text为登录，调节字的格式，大小

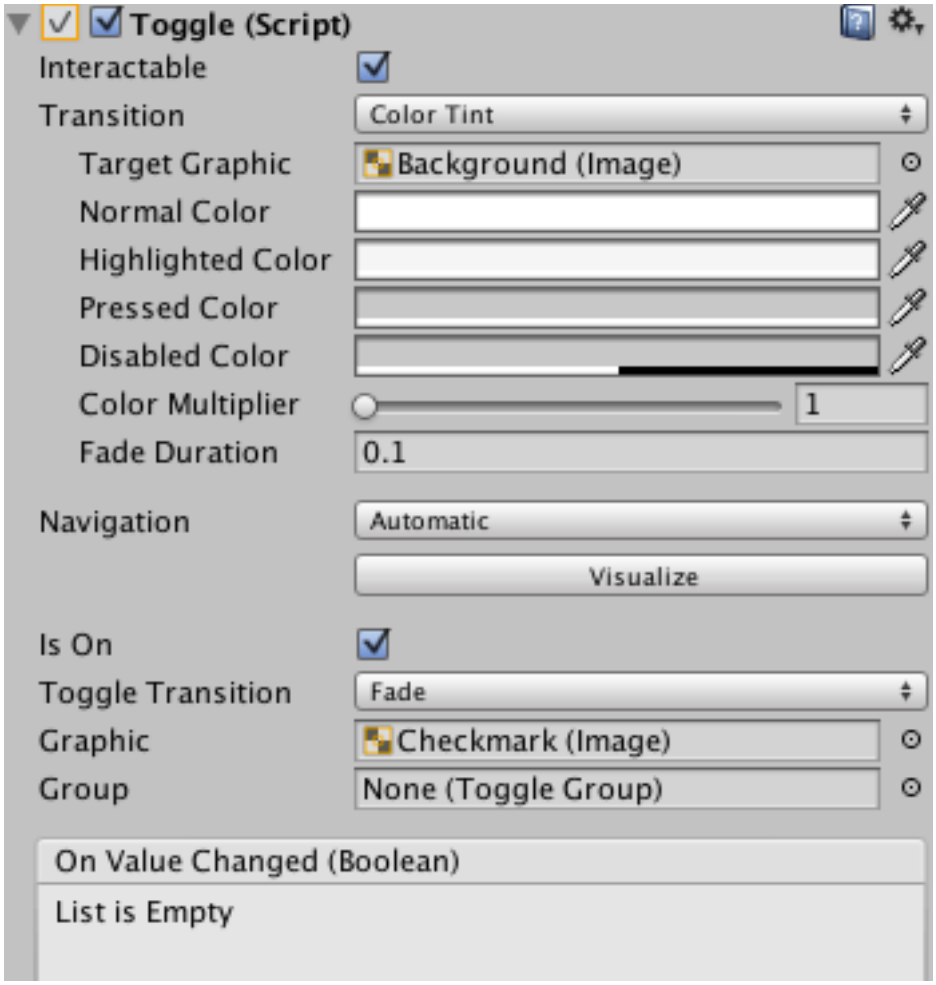
创建脚本，代码如下

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class GameController : MonoBehaviour
{
    public InputField userName;
    public InputField passWord;
    public Text showMessage;
    public void OnLogonButtonClick()
    {
        string userName = this.userName.text;
        string passWord = this.passWord.text;
        if (userName=="admin"&&passWord=="admin")
        {
            print("登陆成功！");
        }
        else
        {
            showMessage.gameObject.SetActive(true);
            showMessage.text = "您的用户名或密码错误,请重新输入！";
            StartCoroutine(DisappearMessage());
        }
    }
    IEnumerator DisappearMessage()
    {
        yield return new WaitForSeconds(1);
        showMessage.gameObject.SetActive(false);
    }
}
```

三：UGUI高级控件

3.1 Toggle控件

Toggle，根据字面意思，可以直接将其翻译为开关控件，其由



Label，Image，Toggle组件组成。其中Label与Image控制组件的外观，Toggle控制组件的事件与行为。

其中IS ON确定开关的状态，默认勾选。添加事件的方法与Button类似。

如何定义一个开关组？

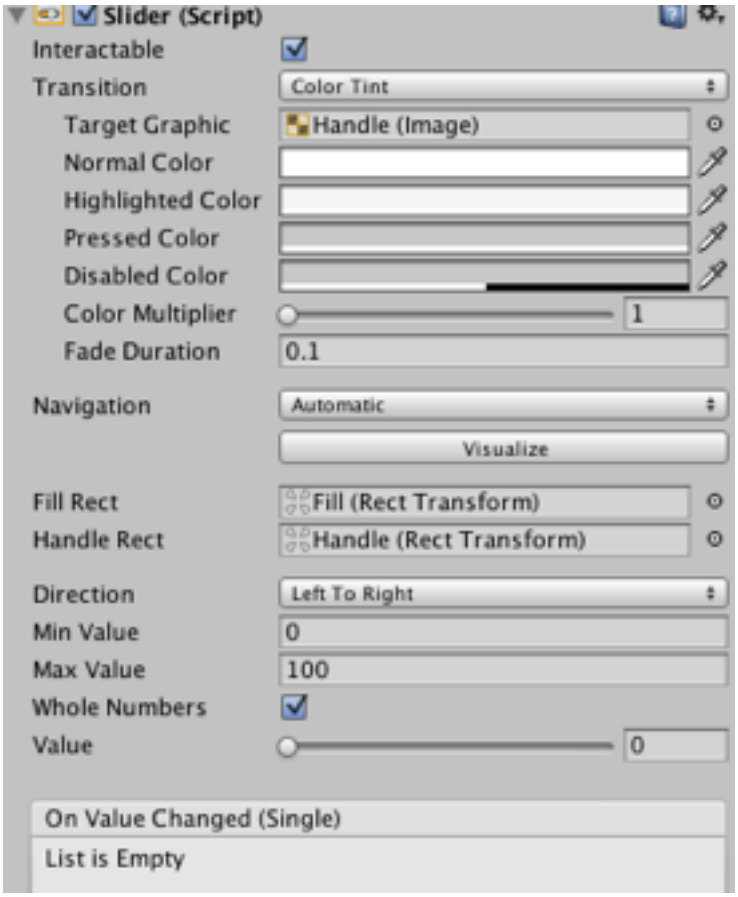
（1）定义一个ToggleGroup空对象，添加Toggle Group脚本组件

（2）选择所有需要组的Toggle控件，在对应的Group属性中拖入刚才创建的ToggleGroup

（3）为了控件显示的清晰，建议把所有的Toggle控件设置为ToggleGroup对象的子对象。

3.2 Slider控件

Slider主要用于游戏中设置音量的大小，Slider常见的属性



如图。属性中Value是当前的滑块数值，WholeNumbers是整数数值。

3.3 ScrollBar控件

此控件基础的属性在课上进行详细讲解。

3.4 DropDown控件

此控件比较实用，主要用于设计下拉列表，在代码中获取其value即可。

3.5 Panel控件

Panel控件是UGUI界面的容器类控件，能够管理一批子节点。比较实用，可以进行常见的各种设置。

3.6 ScrollRect复合控件

ScrollRect不属于UGUI的内置控件，而是依据官方内置的进行二次开发。

(1)新建一个场景，创建一个Image控件，命名为ScrollRect，这个图像主要负责显示大量子控件。按照要求添加ScrollRect组件。

（2）在ScrollRect下面创建一个空的对象“Content”，以及空对象的子对象若干个Button控件，同时设置空对象的width和height能够容纳所有的Button。

（3）创建ScrollBar控件，调整位置。

（4）Scrollbar控件赋值给ScrollRect组件的HorizontalScrollbar属性。Content空对象赋值给ScrollRect组件的Content属性。测试程序。

（5）给ScrollRect添加Mask组件，运行程序即可得到该控件

3.6 Button动画设置

Button在设置的时候，使用Animation是Unity的一个特色。使用动画状态机可以对不同状态下按钮的位置，大小，旋转，图片等大参数进行设置。

（1）首先创建一个Button，修改Button组件的Transition选为Animation，然后单击下方的Auto Generate Animation按钮，在弹出的对话框中保存动画控制器。

（2）创建好之后，选择Window—>Animation打开动画编辑器窗口，然后单击Hierarchy面板上的Button控件，在Animation左上角下拉框选择想要编辑的按钮状态。

（3）我们在例子中需要单击按钮之后进行弹性缩放，所以将当前的按钮状态选择为Pressed，然后单击AddProperty按钮，在展开的RectTransform中单击Scale右边的十按钮。

（4）单击下方的Curve进入曲线编辑模式，在该模式下对x,y,z进行设置。

（5）动画编辑结束，关闭Animation窗口，运行场景，当玩家单击按钮时候，按钮就会进行弹性缩放。

```
public class MainMenu : MonoBehaviour {
    // Use this for initialization
    void Start () {
        List<string> btnsName = new List<string>();
        btnsName.Add("BtnPlay");
        btnsName.Add("BtnShop");
        btnsName.Add("BtnLeaderboards");
        foreach(string btnName in btnsName)
        {
            GameObject btnObj = GameObject.Find(btnName);
            Button btn = btnObj.GetComponent<Button>();
            btn.onClick.AddListener(delegate() {
                this.OnClick(btnObj);
            });
        }
    }
    public void OnClick(GameObject sender)
    {
        switch (sender.name)
        {
        case "BtnPlay":
            Debug.Log("BtnPlay");
            break;
```

```
            case "BtnShop":
                Debug.Log("BtnShop");
                break;
            case "BtnLeaderboards":
                Debug.Log("BtnLeaderboards");
                break;
            default:
                Debug.Log("none");
                break;
            }
        }

        // Update is called once per frame
        void Update () {

        }
}
```

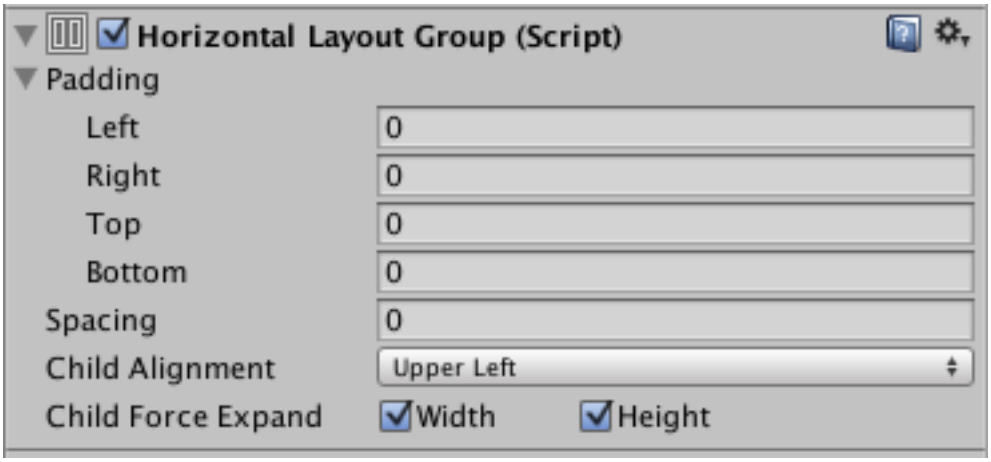
3.7 UGUI布局管理

假设游戏的奖励窗口，由于我们事先不知道获得奖励的数量，但是依旧需要让获得的奖励道具按照一定的布局整齐的出现在界面中，这就需要使用布局的知识。

Unity自带的布局模式分为Horizontal Layout Group、Vertical Layout Group、Grid Layout Group，分别为水平布局，垂直布局，网格布局。接下来我们依次介绍。

1:水平布局

新建一个UIMain空对象，添加Horizontal Layout Group组件，为该对象添加一个水平布局管理器。在该组件的作用下，UIMain的子对象将按照一定的要求进行水平排列。



Padding:布局的边缘填充

Spacing: 布局内的元素间距

ChildAlignment: 对齐方式

ChildForce Expand:自适应宽和高

2:VerticalLayout Group

新建一个UIMain空对象，添加Vertical Layout Group组件，为该对象添加一个垂直布局管理器。在该组件的作用下，UIMain的子对象将按照一定的要求进行垂直排列。内部参数和水平的一样，不再重复。

3:Grid Layout Group

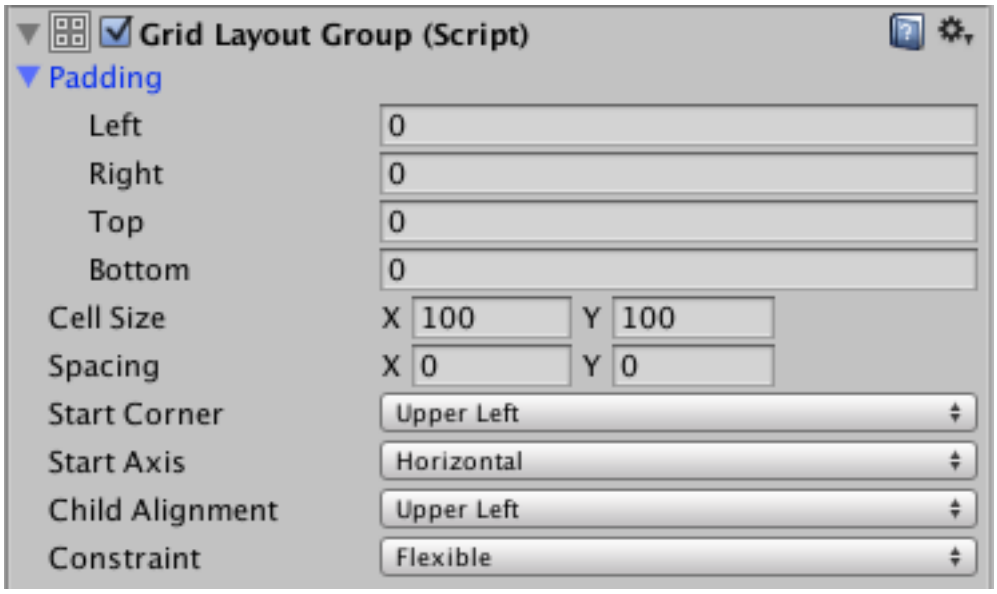
GridLayoutGroup是网格布局管理器，这个组件会自动的将其管理下的UI元素进行网格排列，实现了自动换行的功能，常见于游戏的背包。

Padding: 偏移

CellSize: 内部元素的大小

Spacing:水平间距和垂直间距

StartCorner:第一个元素的位置



StartAxis: 元素的主轴线

Horizontal: 填满一行用一个新行

Vertical: 填满一列用一个新列

ChildAlignment: 对齐方式

Constraint:指定网格布局的行或者列。

我们依次修改其中的属性进行界面的查看。

```
public class NewBehaviourScript : MonoBehaviour {
    public GameObject UIMain;
    public GameObject items;
    // Use this for initialization
    void Start () {
        GameObject item = (GameObject)Instantiate
(items);
        item.transform.SetParent
(UIMain.transform,false);
    }

    // Update is called once per frame
    void Update () {

    }
}
```

在Start方法中实例化出一个新的items预制件，将其设置为挂载有布局管理器组件的UIMain的子对象，然后观察场景，就会发现新的物体已经被排列好了，这在开发游戏中非常方便，不需要考虑排布问题了。

3.8 UGUI不规则形状按钮的碰撞检测

UGUI中自带的按钮是标准的矩形，虽然玩家可以任意换图，但是碰撞区域始终是矩形的。有些时候，可能会用到特殊形状的按钮，其碰撞区域肯定也要符合按钮形状。本节中，我们将创建一个不规则的按钮。

（1）首先创建一个Button控件，可以删除内部的text组件。然后添加Polygon Colloder 2D多边形碰撞组件。

（2）单击EditCollider，在Scene中勾选出想要的碰撞形状

（3）接下来要实现按钮的碰撞区域和PolygonColloder2D区域挂钩，这一步需要重写Image类，新建一个C#脚本，将其命名为UGUIImagePlus.cs,代码如下

```
public class UGUIImagePlus : Image {
     PolygonCollider2D collider;
    void Awake()
     {
          collider = GetComponent<PolygonCollider2D>();
     }
     override public bool IsRaycastLocationValid(Vector2 sp,
Camera eventCamera)
     {
          bool inside = collider.OverlapPoint (sp);
          return inside;
          //return ContainsPoint(collider.points,sp);
     }
}
```

(4)接下来将Button上面的Image移除，同时添加刚才我们创建的UGUIImagePlus组件

（5）挂载之后，进行Button的事件监听，就可以正常的进行指定区域的触摸了。

四：Anchor锚点与屏幕自适应系统

每个控件都有一个Anchor的属性，其作用是当改变屏幕分辨率的时候，当前控件做如何的位置变换，即控件的屏幕自适应特性。

控件的锚点总是相对于自己的上一级来定义的（例如Button中Text中的锚点是不能离开控件本身范围的）中心点在控件外围，就可以旋转控件。控件的4个边沿与Anchor（锚点）的距离是保持不变的，即不受屏幕分辨率变化的影响。

锚点的中心思想：

能够在不同的分辨率下都显示一个正确的位置

中心点相对于锚点的位置永远保持不变

控件的四个边框相对于锚点永远保持不变