# UGUI-Component

## 一、Selectable

UGUI事件响应组件（Button、Toggle...）的基类，UGUI重要的组成部分，是EventSystem的具体接收方。

相关**接口**：

- IMoveHandler：接收通过外设（键盘、手柄等）的方向输入的响应接口
- IPointerXXXHandler：点击/触摸输入的响应接口
- ISelectHandler：当该物体被选中时的响应接口，取决于EventSystem的m_CurrentSelectd
- IDeselectHandler：当该物体取消选中时的响应接口，取决于EventSystem的m_CurrentSelected

输入事件都是基于对象被响应的（例如：点击按钮、拖拽物体），必须有物体，所以可以触发事件的物体的基本条件就是"可选中"，以Selectable为基类可以满足广大操作受众。

**SelectHandler Selectable的基础事件**：由输入检测（InputModule）与输入响应模块（Selectable等组件）调用来更换EventSystem当前对象时被处触发的事件类型。

`ExecuteEvents.selectHandler`

### Selectable生命周期

- Enable：初始化过程会向一个静态Selectable数组中添加自己，这是为IMoveHandler服务。当接收到方向键移动事件时，在导航（Navigation）失败后根据输入方向和当前Selectable组件的坐标会遍历List寻找，最终找到满足条件的新目标Selectable组件。

```
protected override void OnEnable()
{
    //Check to avoid multiple OnEnable() calls for each selectable
    if (m_EnableCalled)
        return;

    base.OnEnable();

    // 数组扩容
    if (s_SelectableCount == s_Selectables.Length)
    {
        Selectable[] temp = new Selectable[s_Selectables.Length * 2];
        Array.Copy(s_Selectables, temp, s_Selectables.Length);
        s_Selectables = temp;
    }
    m_CurrentIndex = s_SelectableCount;
    // 在数组中添加自己
    s_Selectables[m_CurrentIndex] = this;
    s_SelectableCount++;
    isPointerDown = false;
    // 状态变化以及过渡
    DoStateTransition(currentSelectionState, true);

    m_EnableCalled = true;
```

```
}
```

- Disable：将自己从链表中移除，并清除自身状态（存在过渡）。

Selectable通常处理两件事：**导航（Navigation）选择目标**与**状态过渡（Transition）**。

### Navigation

导航有五种类型可供选择：

- None： = 0，无导航
- Horizontal： = 1， 自动地水平方向导航
- Vertical：= 2，自动得垂直方向导航
- Automatic：= 3，自动两个维度地导航
- Explicit：= 4，自定义各个方向地导航目标

**1. 当无导航时：不会成为别的导航目标，自身也不会导航其他物体**

**2. 当处于水平、垂直、二维导航时：目标会根据距离来确定导航上下左右的导航目标（水平只有左右，垂直只有上下）**

**3. 当处于自定义时：可以自己选择上下左右所导航的目标**

当响应OnMove事件时，会根据导航模式寻找对应方向的目标

```
public virtual void OnMove(AxisEventData eventData)
{
    switch (eventData.moveDir)
    {
        case MoveDirection.Right:
            Navigate(eventData, FindSelectableOnRight());
            break;

        case MoveDirection.Up:
            Navigate(eventData, FindSelectableOnUp());
            break;

        case MoveDirection.Left:
            Navigate(eventData, FindSelectableOnLeft());
            break;

        case MoveDirection.Down:
            Navigate(eventData, FindSelectableOnDown());
            break;
    }
}
```

```
public virtual Selectable FindSelectableOnRight()
{
    if (m_Navigation.mode == Navigation.Mode.Explicit)
    {
        // 若导航是自定义，则直接放回设置的右侧目标
        return m_Navigation.selectOnRight;
```

```
    }
    if ((m_Navigation.mode & Navigation.Mode.Horizontal) != 0)
    {
        // 导航满足方向时，根据世界坐标遍历List<Selectable>寻找下一个可选择对象
        return FindSelectable(transform.rotation * Vector3.right);
    }
    return null;
}
```

### Transition

状态过渡有四种类型可供选择：

- None：无
- ColorTint：颜色变化
- SpriteSwap：Sprite图像变化
- Animation：动画变化

当Selectable状态改变时，会改变其状态相关属性（颜色、图像、动画），并执行对应过渡。

```
protected virtual void DoStateTransition(SelectionState state, bool instant)
{
    if (!gameObject.activeInHierarchy)
        return;

    Color tintColor;
    Sprite transitionSprite;
    string triggerName;

    ... // 根据类型的属性赋值

    switch (m_Transition)
    {
        case Transition.ColorTint:
            StartColorTween(tintColor * m_Colors.colorMultiplier, instant);
            break;
        case Transition.SpriteSwap:
            DoSpriteSwap(transitionSprite);
            break;
        case Transition.Animation:
            TriggerAnimation(triggerName);
            break;
    }
}
```

- **颜色变化**：通过Graphic中的CrossFadeColor方法改变颜色，是通过TweenRunner来实现协程动画的。

```
void StartColorTween(Color targetColor, bool instant)
{
    if (m_TargetGraphic == null)
        return;

    m_TargetGraphic.CrossFadeColor(targetColor, instant ? 0f :
m_Colors.fadeDuration, true, true);
}
```

- **图像变化**：将新图像设置给Image

```
void DoSpriteSwap(Sprite newSprite)
{
    if (image == null)
        return;

    image.overrideSprite = newSprite;
}
```

- **动画变化**：执行Animator的Trigger操作

```
void TriggerAnimation(string triggername)
{
    if (transition != Transition.Animation || animator == null ||
!animator.isActiveAndEnabled || !animator.hasBoundPlayables ||
string.IsNullOrEmpty(triggername))
        return;

    animator.ResetTrigger(m_AnimationTriggers.normalTrigger);
    animator.ResetTrigger(m_AnimationTriggers.highlightedTrigger);
    animator.ResetTrigger(m_AnimationTriggers.pressedTrigger);
    animator.ResetTrigger(m_AnimationTriggers.selectedTrigger);
    animator.ResetTrigger(m_AnimationTriggers.disabledTrigger);

    animator.SetTrigger(triggername);
}
```

#### 事件响应

Selectable对事件的响应并没有做太多的处理，仅仅只是为它的子类们做好了通用部分的处理（改变状态并执行对应过渡），具体对事件的处理交给了它的子类们。

## 二、Component

### 1. Image & RawImage

### 2. Text & Shadow & Outline

**3. Button**

**4. Toggle**

**5. Slider**

**6. Scrollbar & ScrollRect**

**7. Dropdown**

**8. InputField**