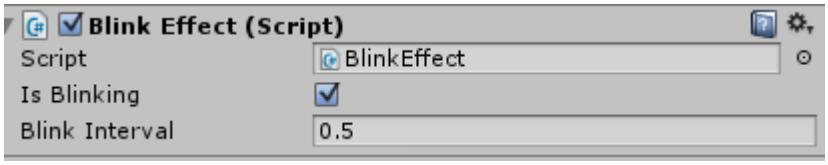
Unity 4.6 UI Addons Guide

This document contains detailed instructions and code snippets for all UI Addons for Unity 4.6 UI System.

Blink Effect -> You need to attach the script (BlinkEffect)to your UI element.



IsBlinking - if true button is blinking.

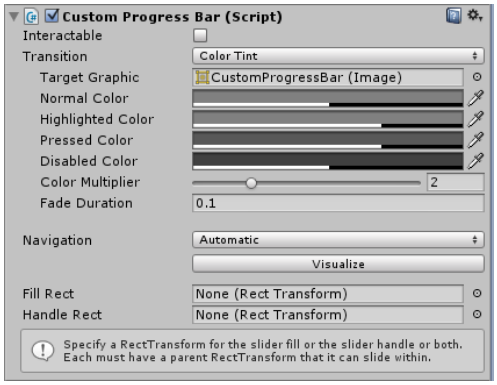
Blink interval - time between every blink

Both properties could be changed at runtime also.

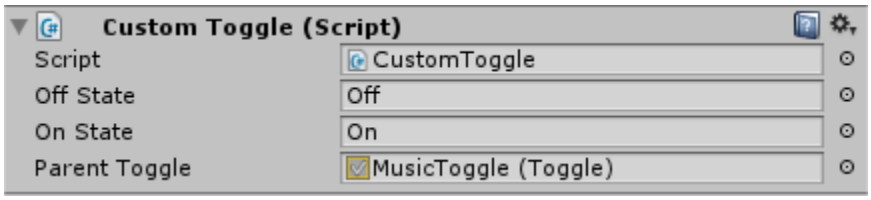
There is already added prefab for you to drag.

Custom Progress bar - This class inherits the new UI Silder. Instead of handle and fill object is uses boxes (UI images) which are child objects of a grid. You can add or remove the number of children and the progress bar will still work as expected. If you need more detailed progress bar you can use more children otherwise just use few one. Active and inactive box are separated by color. Both active and inactive colors are public variables of the CustomProgressBar script.

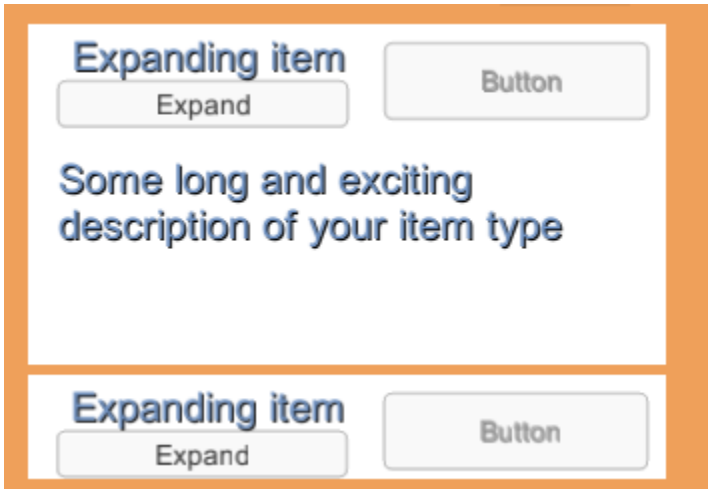
Note - Fill and Handle rects must be left None as in the Prefab showed in the example scene.



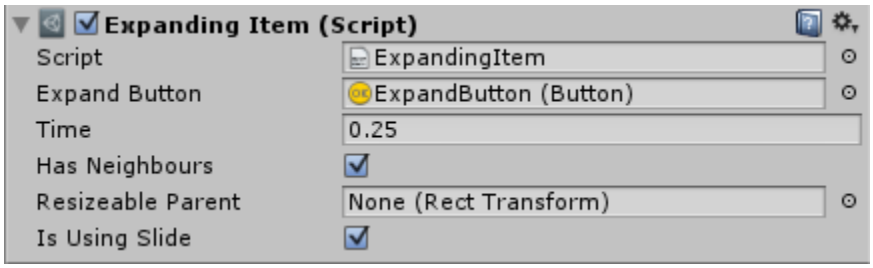
Custom Toggle - Attach the script CustomToggle to the UI element. Custom Toggle instead of enabling additional gameobject(UI image) when checked it just switches between two images (On and OFF states)Every state is gameobject(UI Image) that is a child of the toggle. You can leave the fields of the ON and Off states in the CustomToggle script empty. Then the script will automatically assign them following the logic that First Child of the toggle will be OFF state and second child will be On state. There is added prefab that is showed in example scene.



Expanding Item



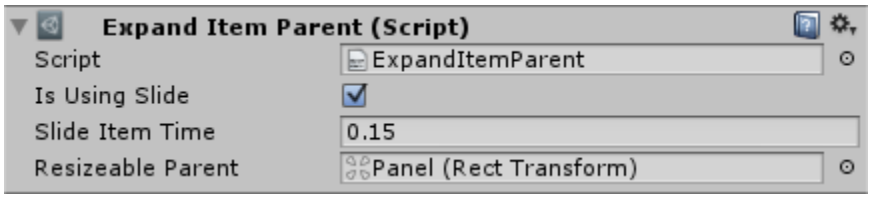
UI element that uses Mask component to create expanding effect. There is already made prefab ready to be customized for your game.



Has Neighbours - If your expanding item is child with other expanding items you need to set that flag to true(for all childs). This way if one item is expanding it will tell its neighbours and they will move down or if the item is folding back to original state it will make the rest of the items to move accordingly. All children will move in a manner dictated by their order in the hierarchy view - the order in the hierarchy must match the Y axis order in the scene (which item is above which other...). If you choose IsUsingSlide then expanding and folding process will be smooth else it will be instant. If the item has neighbours then you need to set the IsUsingSlide flag from the parent object. Parent object of expanding items must have ExpandItemParent added. If it's not added ExpandingItem will check at Awake and add it if needed. Check "ExpandingItemParent" prefab at the example scene.

Resizeable Parent - If you drag RectTransform there it will be resized without affecting all childrens postition and size on expand event and resized back to its original size on fold event. It is usefull when used in scroller. Check scene SceneTwo from the unity package.

Note. If you resize for some reason expanding item then you need to call the public method InitExpandData() to recalculate all variables needed to smoothly move the borders of the mask component.



There is public System.Action for Fold End Event and Expand End Event. You can assign them before clicking on the expand/fold button and they will be called depending on the event type - fold or expand.

```
public Action onFoldEndAction; public Action onExpandEndAction;
```

Fade Effect Script - FadeEffectScript must be added to UI Image(Gameobject with UI Image).

After that you can call one of the methods - `StartFadeOut` or `StartFadeIn` which takes two parameters - time and optionally callback (System.Action) that will

be called at the end of the Fade out/in. Also you can specify delay after which the effect will start.

Pop Effect - PopEffect must be added to UI element of your choice. It has two main modes - Pop Up and Pop Down. Both effects are called like this:

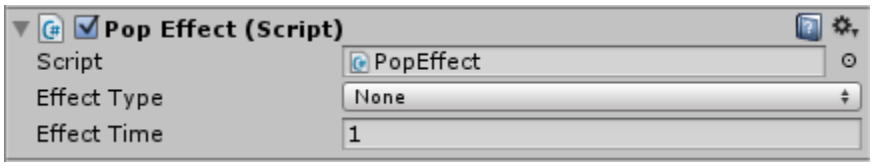
```
StartPopEffect(PopEffectType type, PopEffectEndAction endEffect, Action callback = null, float delay = 0)
```

type - Could be None, PopUp, PopDown.

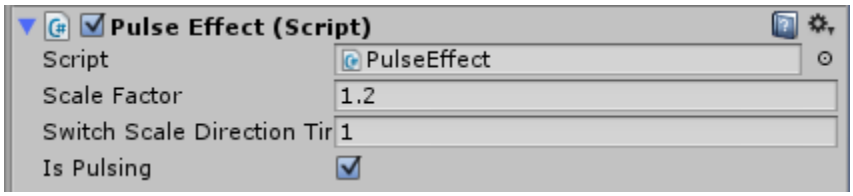
endEffect - Could be None, Deactivate, Destroy, Callback.

callback - if for endEffect is chosen Callback this field(parameter) must assigned.

delay - delay after which the effect will start.



Pulse Effect - PulseEffect must be added to UI element of your choice. By calling StartPulseEffect or StopPulseEffect you control the pulse effect. By assigning scaleFactor before calling the StartPulseEffect you control how bigger will be the UI element at its peak just before start scaling down. By assigning switchScaleDirectionTime you control how much time it's going to take to reach the max size just before it starts scaling down.



Rolling Text Effect - In order to use rolling text effect you need to add the RollingTextEffect to your UI Text. You activated by calling :

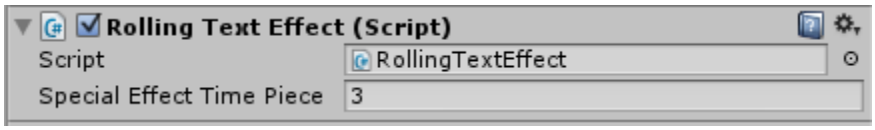
```
public void StartCounting(string text, float time, bool scaleEffect = true, bool isUsingSpecialEffect = true)
```

text/number - Target text that will be visible at the end of the counting.

time - overall time that the counting will take

scaleEffect - whether to use scale effect while counting

isUsingSpecialEffect - Whether to reveal each character/digit in a consequential manner. There is another public field called specialEffectTimePiece that has default value of 0.3f and dictates at which relative point of the counting will the consequential revealing begin. In other words if you pass overall time of 3 seconds and specialEffectTimePiece has the default value of 0.3f then at 2 sec it will start the consequential revealing. Check the example scene and click the "Push Me" button.



Text appear effect - That script animates a string over time so that its chars are shown in time manner. Check SceneTwo. It has one simple method called `public void StartTextAppear(string text, float time, float delay = 0f, EffectEndType endEffectType = EffectEndType.None, Action callback = null)`

text - string you want to display over time

time - overall time that will the animation take

delay - delay before starting the animation

endEffectType - what do you want to happen when the animation is over

callback - if you want to call some method at the end...

UITools - One central Script from which you can call all effects and apply them to UI elements of your choice without the need to add/remove scripts.

From this class you can use all effect dynamically from your script at runtime. Note that you need to add "using UIAddons;" at the beginning of your script. All Available methods are:

`UITools.SetElementAnchor(LocalAnchor localAnchorType, GameObject uiElement)` -> Sets the anchor of the ui element locally. Unity 4.6.x ships with the ability to set the anchor

according the screen space(Ex. bottom right is the bottom right corner of the screen).Here bottom right means bottom right corner of the ui element. Note SetElementAnchor will not work if your canvas scaler is set to Constant Physical Size. If you are using Canvas Scaler type "Scale with screen size" then your preview Game Window should be with the same resolution as entered in the Canvas Scaler. Setting local anchor will work no matter of the parent object but that parent object must have its pivot point set in its local center as all UI elements are added by default.

BlinkEffect UITools.StartBlinking(RectTransform uiElement, float blinkInterval = 0.5f) -> Adds blinking effect and starts it with the given blinkInterval and returns the BlinkEffect instance.

FadeEffectScript UITools.StartFadeEffect(RectTransform uiElement, FadingType type, float time, EffectEndType endEffectType = EffectEndType.None, Action callback = null) -> Starts fade effect on the given object. EnfEffectType is used to point what you want to happen at the end of the effect. By default none.If you choose Callback then you can pass Action at the callback parameter.

PopEffect UITools.StartPopEffect(RectTransform uiElement, float time, PopEffectType type, EffectEndType endEffect, Action callback = null) -> Starts pop effect. EnfEffectType is used to point what you want to happen at the end of the effect. By default none.If you choose Callback then you can pass Action at the callback parameter.

PulseEffect UITools.StartPulseEffect(RectTransform uiElement, float maxScaleFactor = 1.5f, float switchScalingDirectionTime = 1f, float delay = 0) - Adds if needed and starts Pulsing effect.MaxScaleFactor is used to define how big should the element become at its peak.SwitchScalingDirectionTime defines the interval at which the scaling direction will switch.

PulseEffect UITools.StopPulseEffect(RectTransform uiElement, float delay) - If the given element has pulse effect it will stop it and return the instance of it.

RollingTextEffect UITools.StartTextRollingEffect(RectTransform uiElement, string text/ float finalNumber, float time, bool scaleEffect = true, bool isUsingSpecialEffect = true, EffectEndType endEffect = EffectEndType.None, Action callback = null, float delay = 0) - This method add to a given string (Text component) rolling effect. If scaleEffect is true then while rolling it will also scale with pulsing behaviour. If usingSpecialEffect is true then at the end it will reveal every character /number consequentially. You can choose from predefined end actions. If you choose Callback then you can pass Action callback at the end of the method.

FadeEffectScript UITools.StartScreenFadeEffect(float time, FadingType type, Color fadeColor, Canvas mainCanvas = null, EffectEndType endEffectType = EffectEndType.None, Action callback = null) - This method makes the screen fades. Ideal for scene transition. It also has predefined endActions and callback.

UITools.ResizeUIElementAbsolute(RectTransform uiElement, int newWidth, int newHeight, float time = 0f, EffectEndType endEffect = EffectEndType.None, Action callback = null, float delay = 0) - This method is used if you need to resize an UI element to a curtain size while keeping its position. It uses exact size in pixels. You can do it over time or

instant just by leaving the time parameter 0f.You can control the resize direction by relocating the pivot point.

UITools.ResizeUIElementRelative(RectTransform uiElement, float newWidth, float newHeight, float time = 0f, EffectEndType endEffect = EffectEndType.None, Action callback = null, float delay = 0) - This method is used if you need to resize an UI element to a curtain size while keeping its position. It uses relative screen size. You can do it over time or instant just by leaving the time parameter 0f. You can control the resize direction by relocating the pivot point.

MovingItem UITools.MoveUIElement(RectTransform uiElement, Vector2 targetCoordinates, float time, Action Callback, bool removeComponentOnSlideEnd, float delay = 0) This method moves an ui element to target coordinates at a given time. You can pass callback to be called at the end of the movement and/or set removeComponentOnSlideEnd in order to remove the script that actually moves the object when moving is done.

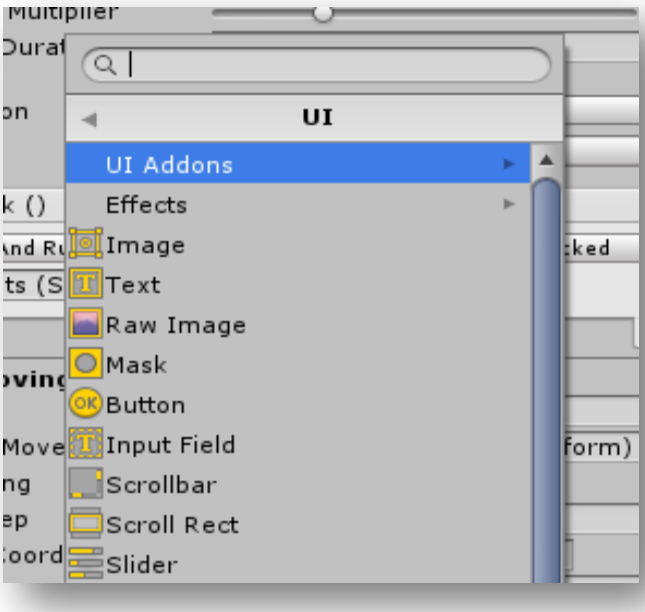
UITools.ResizeUIParentRelative(RectTransform uiParent, int newWidth, int newHeight, float delay = 0) - This method resizes passed RectTransform without affecting the size and position of the children (first level). It takes relative to the screen coordinates (0 - 1). Expanding items has this option embedded. See Expanding items for details.

UITools.ResizeUIParentAbsolute(RectTransform uiParent, int newWidth, int newHeight, float delay = 0) - This method resizes passed RectTransform without affecting the size and position of the children (first level). It takes absolute pixel coordinates.Expanding items has this option embedded. See Expanding items for details.

TextAppearEffect UITools.StartTextAppearEffect(RectTransform uiElement, string text, float time, float delay = 0f, EffectEndType endEffectType = EffectEndType.None, Action callback = null) - This method is adding(if not already attached) Text appear effect script to the passed RectTransform and it will start to animate the given string (text) over a period of time (time parameter). You can set delay before execution, also end effect type and if you need you can pass a callback to be executed at the end. Check SceneTwo of the package for more info.

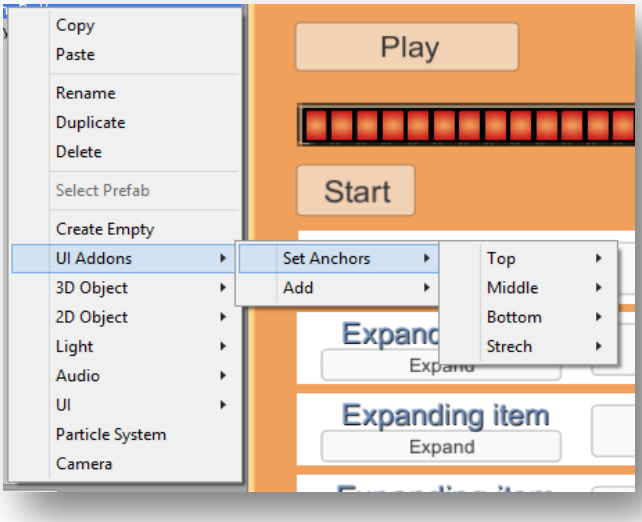
UITools.MakeGrayScale(GameObject uiGameObject, bool applyToChildren = true) - That method will make grayscale the passed uiGameobject by adding Grayscale material. it will also make the font grayscale by using second material just for the fonts. Both materials are public and can be accessed via UITools.GrayScaleMaterial and UITools.GrayScaleFontMaterial. If applyToChildren is true then it will also make grayscale all children recursive.

UITools.RemoveGrayscale (GameObject uiGameObject, bool applyToChildren = true) - That method will remove grayscale material if already attached and will assign none as material to the passed object. Note if you like to have your UI elements with material by default you should consider assigning one after calling that method.



Context Menus

You can add all types of effect and set all types of local anchor via the GameObject context menu.



You can also add the effects by accessing them through the Add Component Menu -> UI-> UI Addons

Note. On some projects with target platform WP8 or Windows Store upon build you may encounter errors related to UIAddons.dll. That's because on this platforms Unity is unable to find the UIAddons.dll file. To fix this simply copy the UIAddons.dll from Plugins/UIAddons/ to your Scripts folder.