KCtrl Help

Hey guys, before we get started,here is my apology:

15 days earlier i began to write scripts,after 7 days all works are done except this doc, now it is still unfinished. For a man never speaks English like me,it is impossible to write a help doc well, so if u get confused or any bugs come out while using KCtrl,just mail me chenxuunity@gmail.com with a Screenshot.

OK, no more nonsense let's start.

What are KCtrl scripts used for?

KCtrls are just sevral scripts which u shall attach to your gameObject to display buttons joysticks and optionbars(let's call them Ctrls) on your screen and scan touch events, if any Ctrls u create within the inspector view is touched, a message will be send to gameObjects which are registed to the root script. If u don't like a message way, 3 status arrays are avilable for u to query status of Ctrls u created.

The most important is KCtrls allow u to assign GUI size by inch rather than pixels in order to unify performance on screens which have different resolutions and sizes. This is very useful on android devices(various resolutions and sizes),

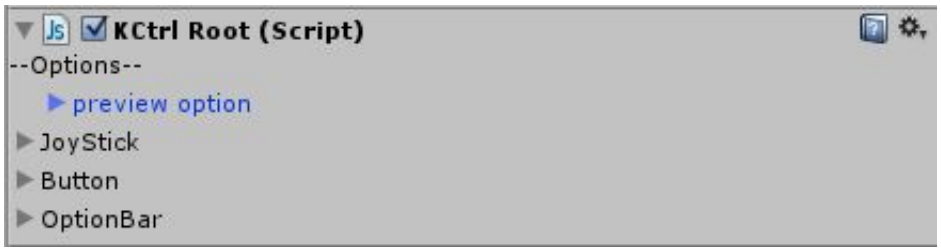
Yes KCtrl is designed for android devices!!!

Setup KCtrls

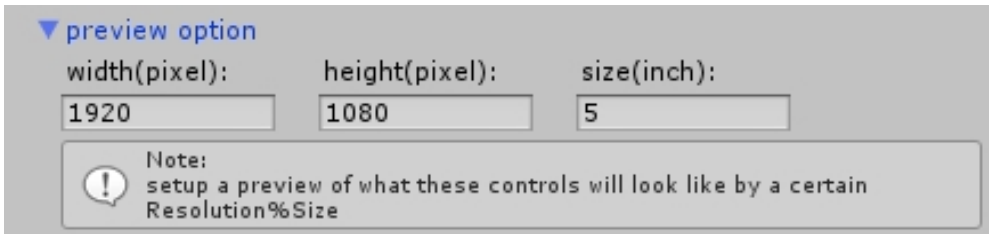
--Create a empty GameObject rename it by any name u like, i prefer 'TestButton'

--Drag the KCtrlRoot.js to GameObject KCtrlRoot just created

--Choose KCtrlRoot in the Hierarchy View, then in the Inspector View u will find KCtrlRoot just like this:

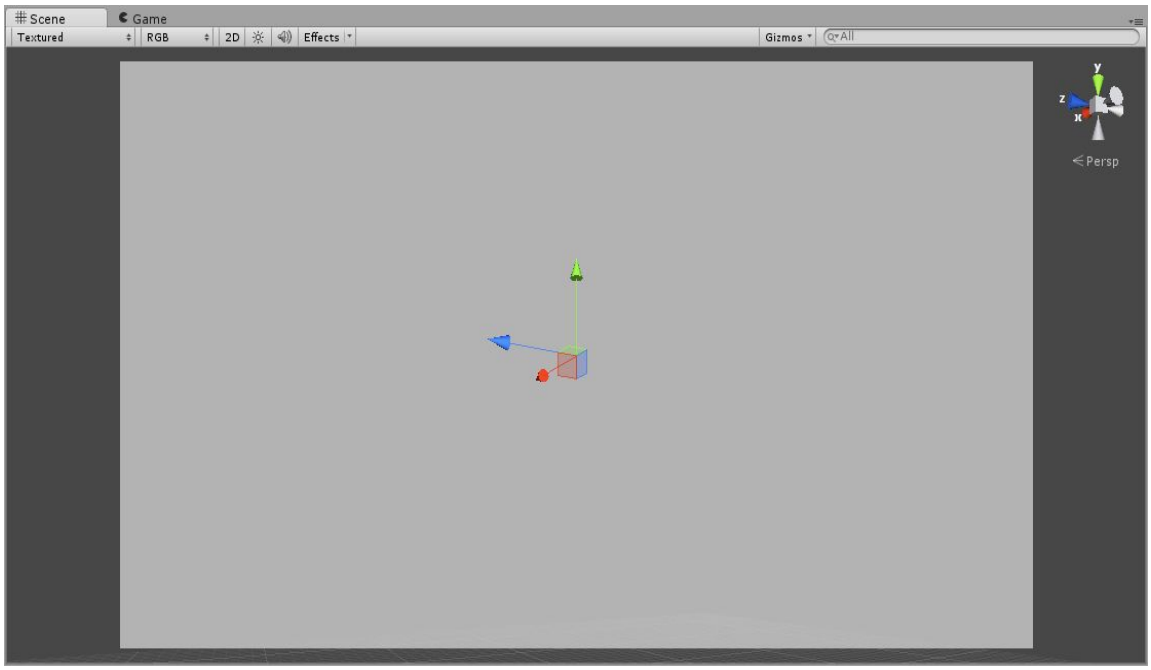


Now we can create Ctrls,but before that **preview option** should be specified



For me it is 1920x1080 resolution and 5 inch screen(don't worry this is just a preview to find out how the Ctrls will look like on a screen of a certain resolution&size, u can specify 800x480 4 inch or any other resolution&size too if u want to know how it looks like) then in the **Scene View** a rect will come out,it is the preview of your

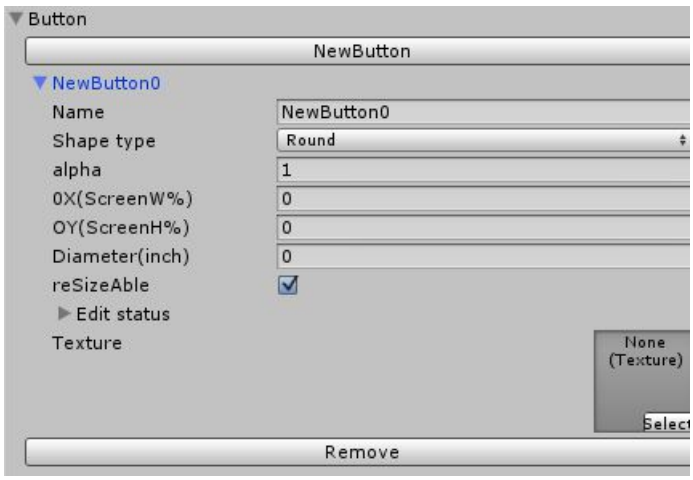
screen all Ctrls will be displayed within it



Button

OK let’s create a button

Click the button NewButton



Well specify these paramenters

Name: name of the Button(used when find this Button should be unique within this script)

ShapeType: shpae of the Button(round/rect/ellipse)

OX/CX: x origin/certer of the Buttonbypercent screen

OY/CY: y origin/certer of the Button by percent screen

Eg: if OX = 50&OY=50 the Button will be displayed in certer of the screen

Alpha: alpha of the Button(0 to 1)

Diameter: Diameter of the Button(ShapeType == round)

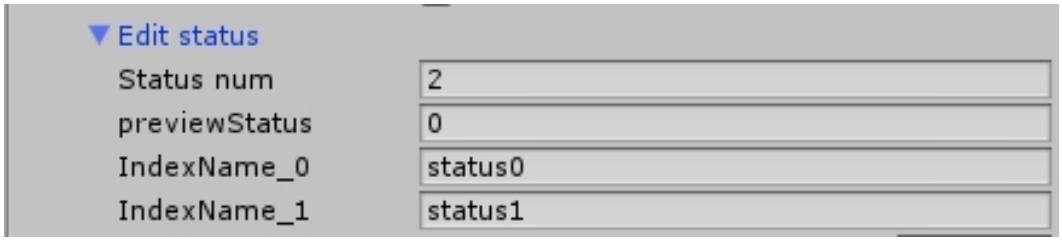
DiameterX/Y: x/y Diameter of the Button(ShapeType == ellipse)

Width/Height:Width/Height of the Button(ShapeType == rectangle)

resizeAble: while this button colides another Ctrl ,if resizeAble is on this button will be resize to fit, otherwise size of this button will not be changed.

Edit Status: a button can has sevral status, it will change to next state after a click

Eg: for a button have 3 status the state will change like this state0->click->state1->click->state1->click->state0



Status num: number of status

PreviewStatus: which state will be displayed in the **Scene View**(0 to **Status num-1**)

IndexName_x: name of the status(should be unique within this button)

Texture:all textures of this button should be combined to one

As i have a texture below:

And the button has 2 status, at the beginning the top half (off) of the textue will be displayed, then after a click the bottom half(on) of the texture will be displayed. If your button has 3 status first top 1/3part diaplay, then middle 1/3 part display, then bottom 1/3 part display, and so on.

Handle touch events :

Here is a class which contains everything we need:

```
class ButtonDataOutput {
	var localName:String; //name of the button whitch this ButtonDataOutput belongs to
	var statusName:String; //name of the current state
	var touchStatus:boolean = false; // true if touched
	var index:int = 0; //index of the current state
	var buttonStatusNum:int = 1; //number of status
	function SetEnable(enable:boolean); //disable message sending
	function GetEnable():boolean;
}
```

And this class extends a class names **MessageManager**

Here it is :

```
class MessageManager{
	/*begin touch*/
	function AddOnBeginMessage(message:String,object:GameObject):boolean
	function DelOnBeginMessage(message:String,object:GameObject):boolean
	function ClearOnBeginMessage():boolean
	/*still touch*/
	function AddOnTouchMessage(message:String,object:GameObject):boolean
	function DelOnTouchMessage(message:String,object:GameObject):boolean
	function ClearOnTouchMessage():boolean
	/*end touch*/
	function AddOnEndMessage(message:String,object:GameObject):boolean
	function DelOnEndMessage(message:String,object:GameObject):boolean
	function ClearOnEndMessage():boolean
}
```

This is not diffcult to understand at all. If u got a ButtonDataOutput object,it is quite easy to know whats going on now, so the question is where can i get it ??!!!Well there are two ways:

First, a massage way:

So if u have a script,whaterver it names, whichever GameObject it is attached to within the current scene, in its Start() function get the instance of your KCtrlRoot.js like this

```
var mRoot :KCtrlRoot = GameObject.Find("TestButton").GetComponent(KCtrlRoot);
```

u can also find it if by a tag if have one,

Then we should use a method of KCtrlRoot.js,like this:

```
mRoot.AddMessage("button","begin","NewButton0","onButtonTouch",gameObject);
```

A message receive function is also needed:

```
function onButtonTouch(object:Object){
    var output:ButtonDataOutput = object;
    Debug.Log(output.localName);
}
```

Now when your button is touched method onButtonTouch will be called,and a ButtonDataOutput object will be delivered as function argument.have a try!!!after that u should see these methods of KCtrlRoot.js:

```
function
AddMessage(ctrlType:String,inputType:String,ctrlName:String,methodName:String,mGameObject:GameObject):boolean

Function
DelMessage(ctrlType:String,inputType:String,ctrlName:String,methodName:String,mGameObject:GameObject):boolean

function ClearMessage(ctrlType:String,inputType:String,ctrlName:String):boolean
```

ctrlType: type of the control sends message could be button/stick/optionbar

inputType: type of the touch(when the message is send begin touch/still touch/end touch) could be begin/touch/end

ctrlName: name of your Ctrl

methodName: name of the message receive method

mGameObject: the **GameObject** which has a script handles input events attached to.

Second a scan way:

There is an array of ButtonDataOutput, it cantains all ButtonDataOutput instances u created to get it, use the method GetButtonData of KCtrlRoot.js like this

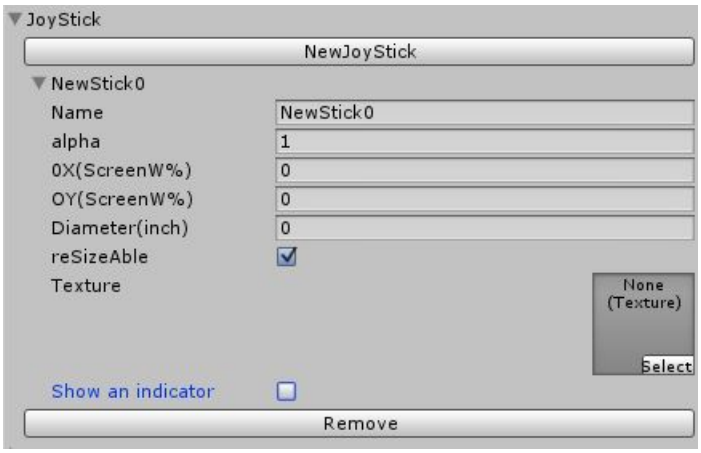
```
buttonArray = mRoot.GetButtonData();
```

find the ButtonDataOutput object your want to use by its localName vriable, then store it , check its status when needed

There is a demo scene names ButtonDemo see to it. By the way all touch status will be updated once per FixedUpdate

JoyStick:

Same create way as Button:



Name: same as button

OX: same as button

OY: same as button

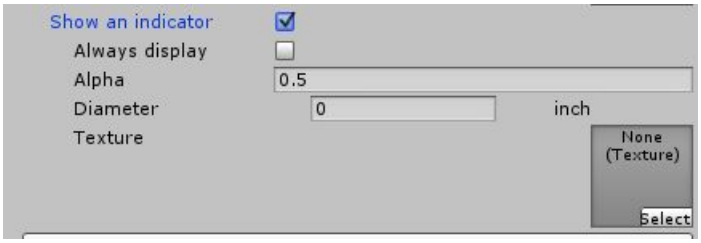
Alpha: same as button

Diameter:same as button

resizeAble: same as button

Texture: texture of your joystick

Show an indicator: sometimes we want to show a texture in the position where touched



Always display: the indicator will display in the origin of this stick when it is not touched if enabled

Alpha: alpha of the indicator

Diameter: diameter of the indicator

Texture: texture of the indicator

Inputs:

Also two ways(message and scan),but the output object is different:

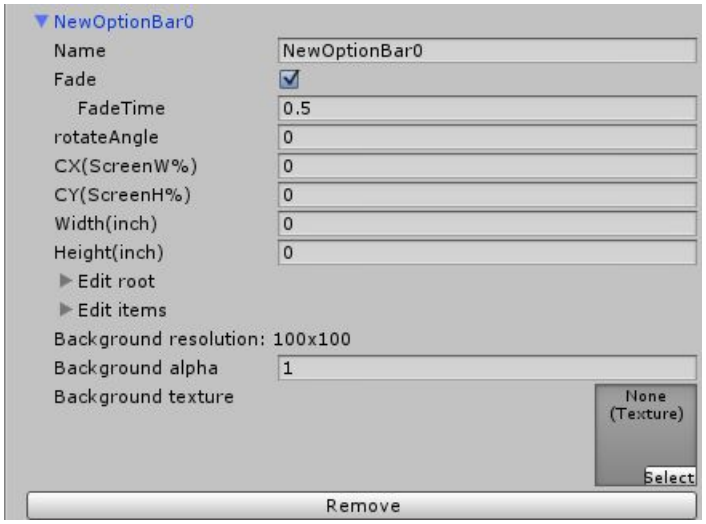
```
class StickDataOutput extends MessageManager{
    //do not change veriables in this class Just read them
    var localName:String; //name of the button whitch this ButtonDataOutput belongs to
    var touchStatus:boolean = false; // true if touched
    var localPosition:Vector2; //position in the stick area calculated by normalize(TouchPosition -
    StickOrigin)
    function SetEnable(enable:boolean); //disable message sending
    function GetEnable():boolean;
}
```

By the way function GetStickData():Array is used to get StickDataOutput array of a KCtrlRoot instance

OptionBar:

I don’t know how doed it call in native English,just call it OptionBar. A OptionBar performs likes a drop down list, it can be rotate and line/ring layouts of subitems are avilable.

Same create way:



Name: name of OptionBar

Fade: will be fade in and out if enabled, fade time can be assigned by second:

`FotateAngle:`angle of rotation around root item’s center(count clock degree)

CX: x origin/certer of the OptionBar by percent screen

CY: y origin/certer of the OptionBar by percent screen

Eg: if OX = 50&OY=50 the OptionBar will be displayed in certer of the screen

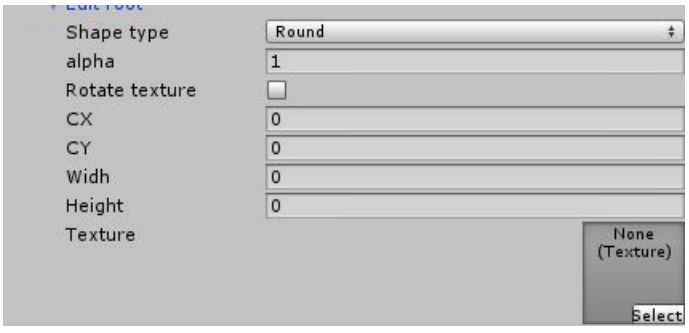
Width/Height:Width/Height of the OptionBar

Background resolution: Background Texture resolution,if no back ground texture is assigned this will be 100x100. All subitems coordinates/size will use the same unit.well how to explain this???? just forget screen coordinates for a while, image the background texture is a canvas, u draw items on it by canvas unit(pixel), then the canvas will be blit to screen , where should the canvas be displayed? u assigned the rect by screen unit(inch)

Background Alpha: alpha of the background (0 to 1)

Background Texture: texture of the background this texture will cover the whole area of OptionBar rect.

Edit root:



Root item is the item that always display if it is touched backgroud and all sub-items will display

Shape type:shpae of the root item (round/rect/ellipse)

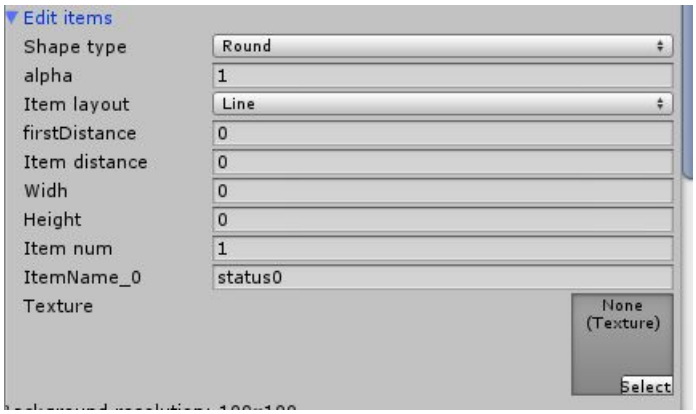
Alpha:alpha of the root item

Rotate texture: whether should the root texture be rotate while rotateAngle is not zero.

CX/CY Width/Height: display area on the background canvas(can be outside of the background still display)

Texture: texture of the root

Edit items:



Shape type:shpae of the sub-items (round/rect/ellipse)

Alpha:alpha of the sub-items

Item layout:

--line:all sub-items will display under the root item

--ring:all sub-items will display around the root item

FirstDistance:how long will the first sun-item move from the root item(move down)

Item distance: distance between sun-items(Item layout == line)

Item angle:angle the sub-items will rotate around the root item from the previous one

Item num: number of sub-items

Itemnae_x: name of sub item should be unique within this OptionBar

Texture: all textures of sub-items in this OptionBar how does it works is the same with button texture

Touch events:

Also two ways.

```
class OptionBarDataOutput extends MessageManager{
	var localName:String; //name of OptinBar this OptionBarDataOutput belongs to
	var itemName:String; name of sub-item last selected
	var index:int = 0; //index of sub-item last selected
	var itemNum:int = 1; //nunber of sun-items
	function SetEnable(enable:boolean); //disable message sending
	function GetEnable():boolean;
}
```

Only begin message could be send, so don't regist other messages(touch/end);

Other words:

To get KCtrl work u just need to manage a KCtrlRoot Component either at runtime or in the I**nspector View.**

if u create a Ctrl in the I**nspector View,** an script will be added to the current GameObject, this may looks wierd if i don't tell u.

There are also some other methods in KCtrlRoot u may use:

```
function SetDisplayEnable(enable:boolean):boolean
function GetDisplayEnable(enable:boolean):boolean
function SetInputEnable(enable:boolean):boolean
function GetInputEnable(enable:boolean):boolean
```

No need to explain what does those are used for ah!