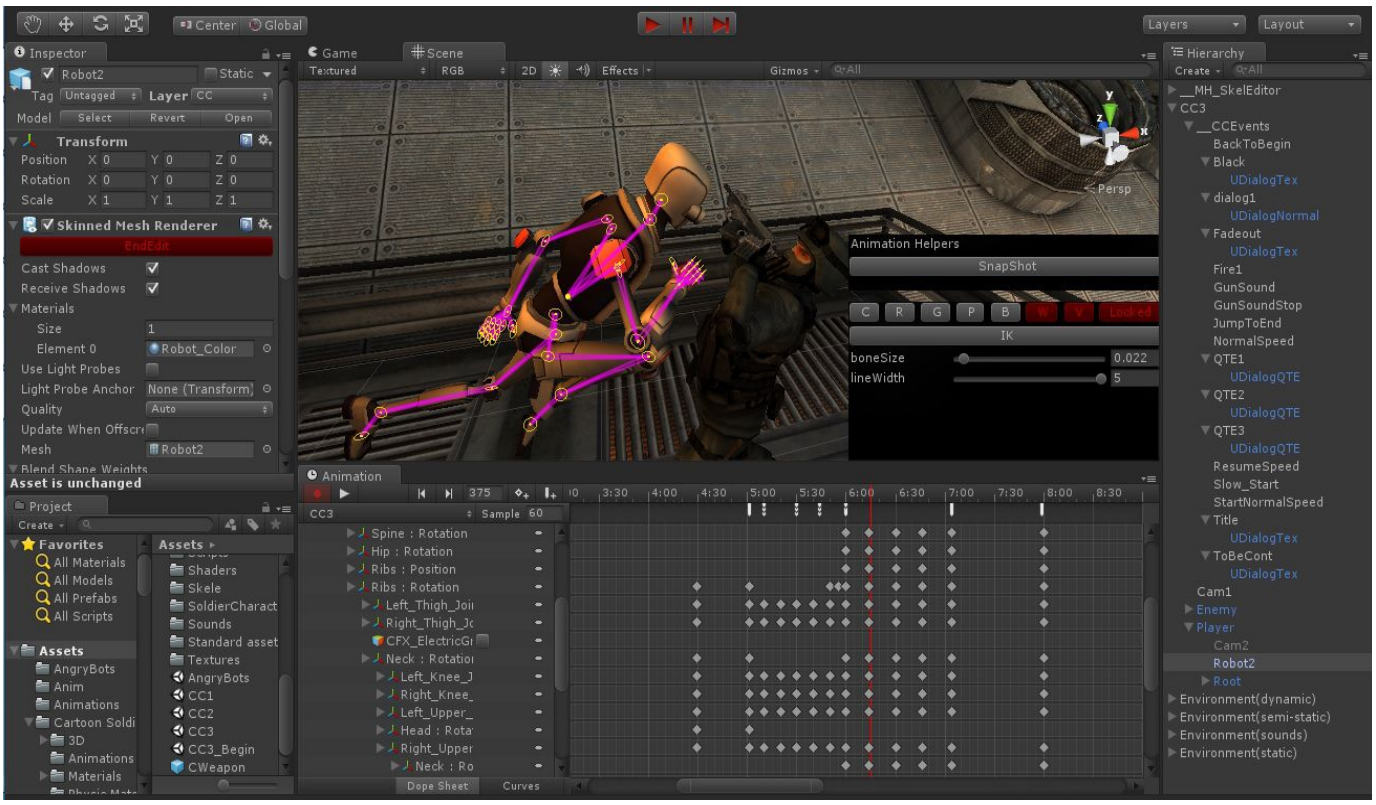
Skele : The CutScene Director

User Manual v0.9

Contents:



● Overview

'Skele: The CutScene Director' is the CutScene Editor Tools included in 'Skele: The Bone Manipulator'.

The Main Aim is to do things in unity editor in ONE place and UNITED way as much as possible.

Just like how we make animations for one character in 'Skele: The Bone Manipulator'(abbr. TBM) , with 'Skele: The CutScene Director'(abbr. TCD) We could make animation for multiple characters simultaneously in the scene.

Developers will be able to easily:

1. Match multiple characters' actions and positions and time.

No need to make a bunch of animations in modeling software, import them into Unity then struggle to match all the animation in cutscene. Just make all the characters animations right within the cutscene in Unity Editor.

2. Match animation with the Fx / Sound / GUI / Trigger / Function-Call / etc.

During game development, there're always so many things need to be combined together. Absolutely, we could make specific editors for each one of them. However it will be messy and there will be a lot of learning cost and maintenance cost.
When work with TCD, everything is clearly put on one same timeline, animator could efficiently match animation with all other stuffs.

3. Control Flow in Cutscene

We could not only play the cutscene through from start to end, but also jump forward / backward in the timeline, which resembles 'jmp' instructions in programming language. With this ability, we could extend a standard cutscene into a better system equipped with QTE, choices, replay, etc.

4. Extend to meet your need

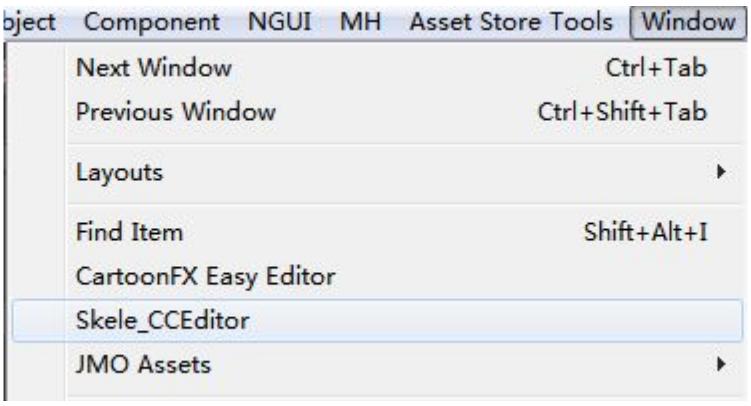
The base structure of TCD is quite easy to understand and extend, if you have new needs for functions / GUI / effects, you could just go ahead and extend it to meet your need.

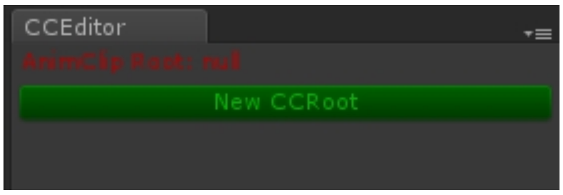
● Get Started

1. Create the Cutscene Controller

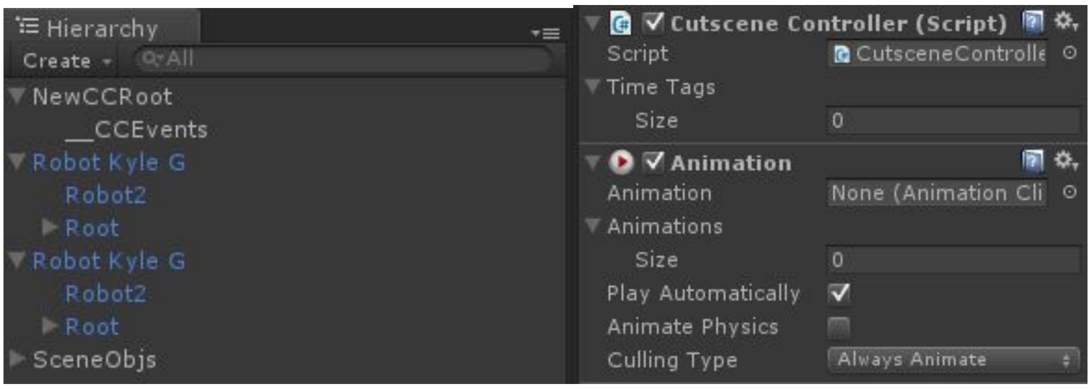
Every <u>Cutscene Controller</u> (abbr. CC) consists of one cutscene.
We need to call up the CCEditor to create a new CC instance.

1) Select the MenuItem ‘Window/Skele_CCEditor’





2) Click the ‘New CCRoot’ Button
It will create a new GameObject (abbr. GO) attached with a CutsceneController and an Animation component.



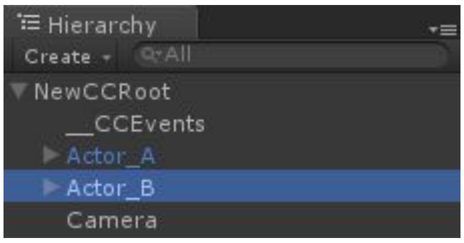
```
Hierarchy
Create ▾  All
▼ NewCCRoot
      __CCEvents
▼ Robot Kyle G
      Robot2
    ▶ Root
▼ Robot Kyle G
      Robot2
    ▶ Root
▶ SceneObjs
```

```
▼ ✓ Cutscene Controller (Script)
  Script              CutsceneControlle
▼ Time Tags
    Size              0
▼ ✓ Animation
  Animation           None (Animation Cli
▼ Animations
    Size              0
  Play Automatically  ✓
  Animate Physics     
  Culling Type        Always Animate
```

2. Prepare the Actors

1) Put the Actors under the CCRoot
[<u>Actor</u> means a GO under CCRoot.]
Just put the actors used by cutscene under the CCRoot. If you don’t want some of them be visible at the start, you could just deactivate them.
Tips: If you’re making a cutscene that need to replace actors during runtime, the <u>inner</u>

actors (the actor which is under CCRoot and will be replaced during runtime) must share skeleton with the outer actors (the actors which will replace inner actors during runtime)



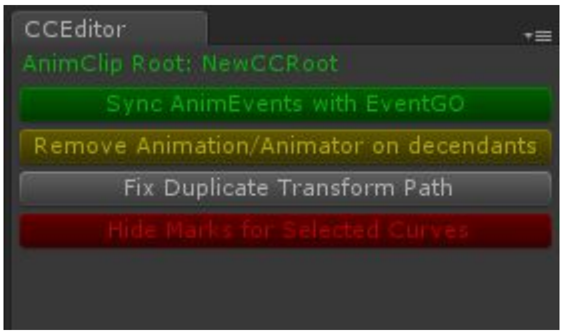
2) Process the Actors

The animation we will make should be stored in the Animation component of CCRoot.

However if any actor under CCRoot has Animator/Animation component, it would interfere the cutscene editing work. We could click the 'Remove Animation/Animator on decendants' button to automatically remove the components.

Also, Unity Animation Window(abbr. UAW) doesn't distinguish two GOs with same Transform path, we could auto-fix such cases by clicking the 'Fix Duplicate Transform path' button.

[Transform Path: the GO's hierarchy path from root]


CCEditor
AnimClip Root: NewCCRoot
Sync AnimEvents with EventGO
Remove Animation/Animator on decendants
Fix Duplicate Transform Path
Hide Marks for Selected Curves

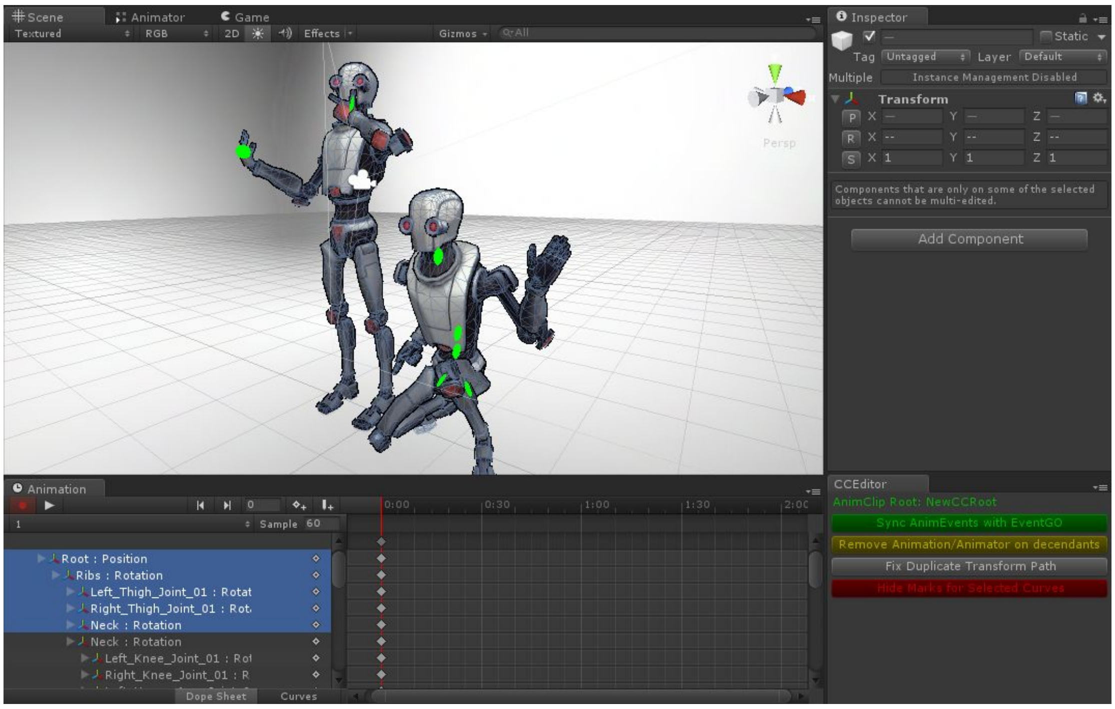
3. Make the Cutscene

Now we're prepared to make the cutscene.

If you're not familiar how to use UAW(Unity Animation Window), you could check out the Animation View Guide in Unity Reference.

You could find out more info about how to make animation for characters with *Skele* in the manual of TBM (The Bone Manipulator).

Tips: You should keep CCEditor window open during editing cutscene, it will mark the position of transforms of active curves in the UAW's dope sheet; This is especially useful when there're multiple characters with similar skeleton, you will see it when you try it yourself, :)

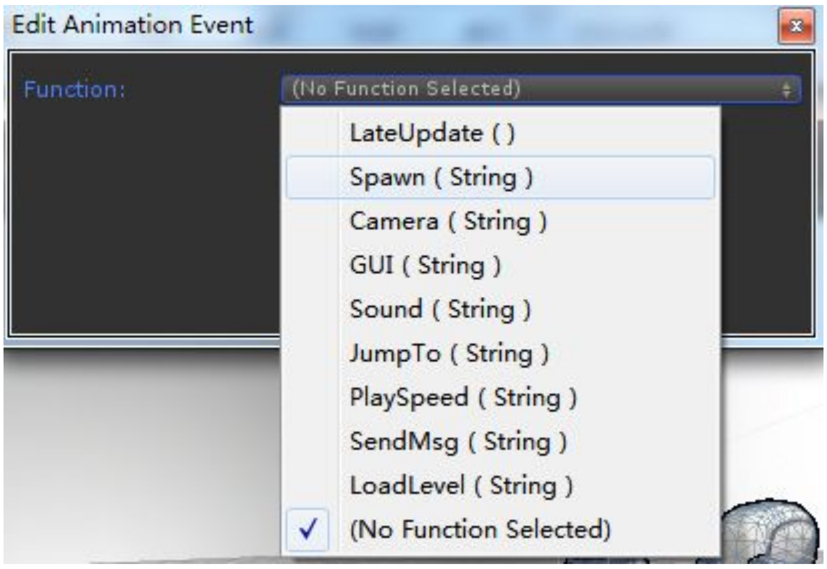

Scene
Animator
Game
Textured
RGB
2D
Effects
Gizmos
All
Persp
Inspector
Static
Tag Untagged
Layer Default
Multiple
Instance Management Disabled
Transform
P X — Y — Z —
R X -- Y -- Z --
S X 1 Y 1 Z 1
Components that are only on some of the selected objects cannot be multi-edited.
Add Component
Animation
0
1
Sample 60
0:00
0:30
1:00
1:30
2:00
Root : Position
Ribs : Rotation
Left_Thigh_Joint_01 : Rotat
Right_Thigh_Joint_01 : Rot.
Neck : Rotation
Neck : Rotation
Left_Knee_Joint_01 : Rot
Right_Knee_Joint_01 : R
Dope Sheet
Curves
CCEditor
AnimClip Root: NewCCRoot
Sync AnimEvents with EventGO
Remove Animation/Animator on decendants
Fix Duplicate Transform Path
Hide Marks for Selected Curves

• Standard Cutscene Components

TCD provides a bunch of common components to help developers go with cutscene editing work, such as GUI / Sound / Spawn / SendMsg / JumpTo / LoadLevel / etc.

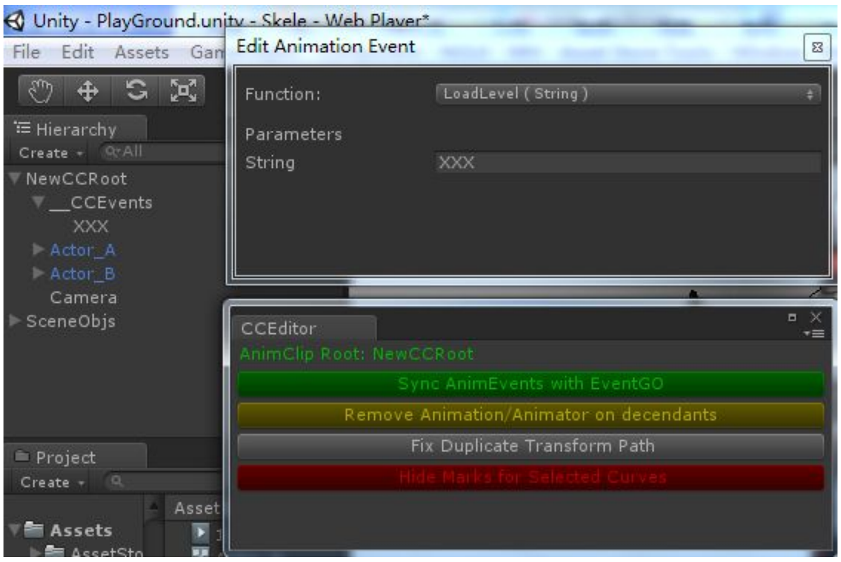
We will go through and introduce this list of components in this section. But please be noted that, it's very easy to extend your own component to meet your specific need. Sometimes, it might be more efficient to add new component than try making up with existing blocks.

To use these components, you just need to use the standard UAW animation event system. (not familiar with how to use Animation Event? Check the Unity Reference *Using Animation Event* section)



1) LoadLevel

LoadLevel is the easiest component, so let's start with it and introduce how to sync animation events with EventGOs.


Unity - PlayGround.unity - Skele - Web Player*

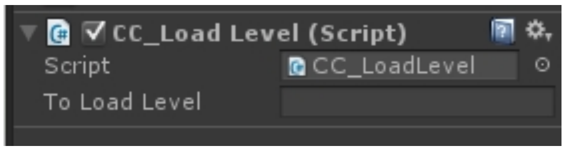
Edit Animation Event

Function: LoadLevel (String)

Parameters

String XXX

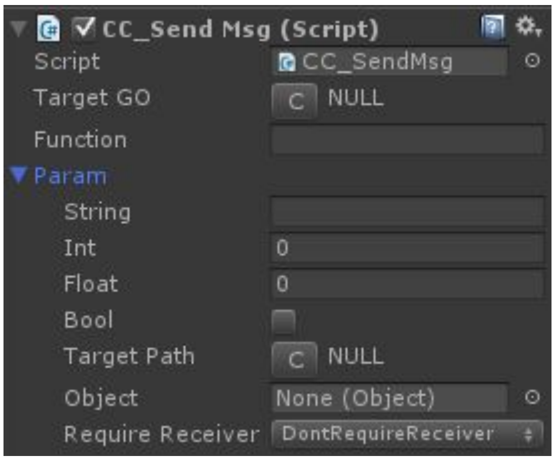
So, after you give the Animation Event a name (any name, just ensure don't have two events with same name), you need to click the 'Sync AnimEvents with EventGO' button, this will automatically create/destroy EventGO with same name under 'CCRoot/__CCEvents' to match with the animation events on the timeline. The newly created GO will be automatically attached necessary component on its creation.



CC_Load Level (Script)
Script: CC_LoadLevel
To Load Level:

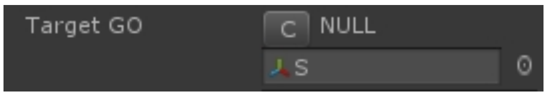
LoadLevel is simple, fill the name of the scene you want to load when cutscene hit the time, and you're done.

2) SendMsg

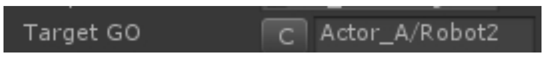


CC_Send Msg (Script)
Script: CC_SendMsg
Target GO: C NULL
Function:
Param
String:
Int: 0
Float: 0
Bool:
Target Path: C NULL
Object: None (Object)
Require Receiver: DontRequireReceiver

SendMsg is simple too, and should be used by programmers mostly, if you are animator and don't quite understand the SendMsg mechanism, you don't have to worry about that.
But there is an important concept we need to introduce here.
Mostly, we don't use direct object reference but transform path instead.
Why? Becuase we want to replace the actors during runtime, and maintain the reference would cause trouble.
But no worry, you don't have to fill in the string by hand, instead, you could just click the "C" button there, and choose the GO you need, the transform path is filled automatically.



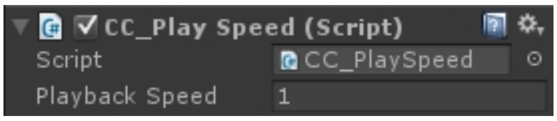
Target GO: C NULL
S



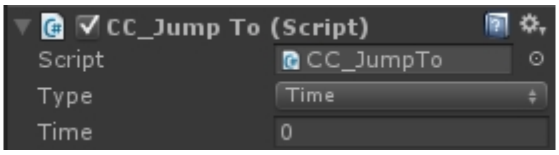
Target GO: C Actor_A/Robot2

The 'Param' field resembles the structure of AnimationEvent, except replace the object reference with transform path.

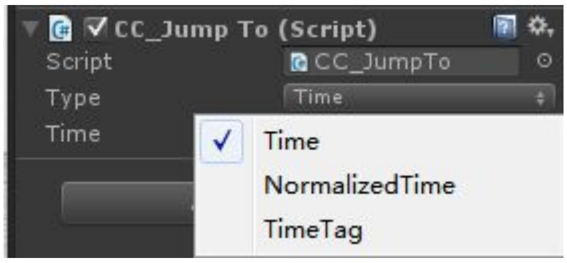
3) PlaySpeed
PlaySpeed will change the current game running speed, you could make slow motion for your cutscene with it.
1 means normal speed, 0 will stop completely.
(technically , it changes Time.timesScale)


CC_Play Speed (Script)
Script CC_PlaySpeed
Playback Speed 1

4) JumpTo
JumpTo enables you to jump the cutscene to specified time, it's simple and powerful when used correctly, you could use it to lead your cutscene to different segments for different dialog options, QTE, etc.

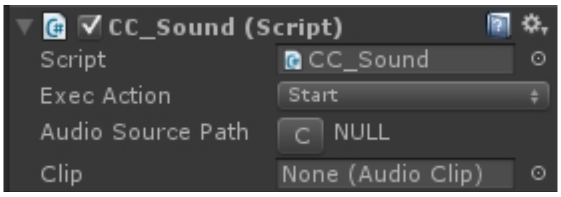

CC_Jump To (Script)
Script CC_JumpTo
Type Time
Time 0

You could specify three kinds of value here,
Time: specify an absolute value on timeline, e.g.: 10sec, 1.5sec, etc.
NormalizedTime: specify a value between 0 and 1
TimeTag: specify a string which is mapped to an absolute time in CC;


CC_Jump To (Script)
Script CC_JumpTo
Type Time
Time

Tips: TimeTag is useful, consider you have a gunshot animation at 10sec position, and have 10 places jump to the gunshot place. If you don't use TimeTag, when you move the gunshot animation to 11sec position, you will have to change all the 10 places' jump setting from 10s to 11s; But if you do use TimeTag, you just need to remap the tag's time from 10s to 11s, and all is done.

5) Sound
Sound component enables us to start/stop audio clip on specified AudioSource.
It's required to specify an AudioSource, the Clip is optional.
If you try to start an audio clip on AudioSource and a clip is given, it will replace the original clip on the AudioSource.


CC_Sound (Script)

Script: CC_Sound

Exec Action: Start

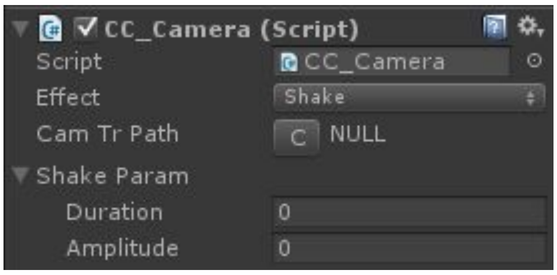
Audio Source Path: NULL

Clip: None (Audio Clip)

6) Camera

Camera component is designed for ‘Camera Shake’, ‘Contrary Color’, and other effects. Well it has only Camera shake for now.

You only need to specify the Camera, shake time and amplitude.


CC_Camera (Script)

Script: CC_Camera

Effect: Shake

Cam Tr Path: NULL

Shake Param

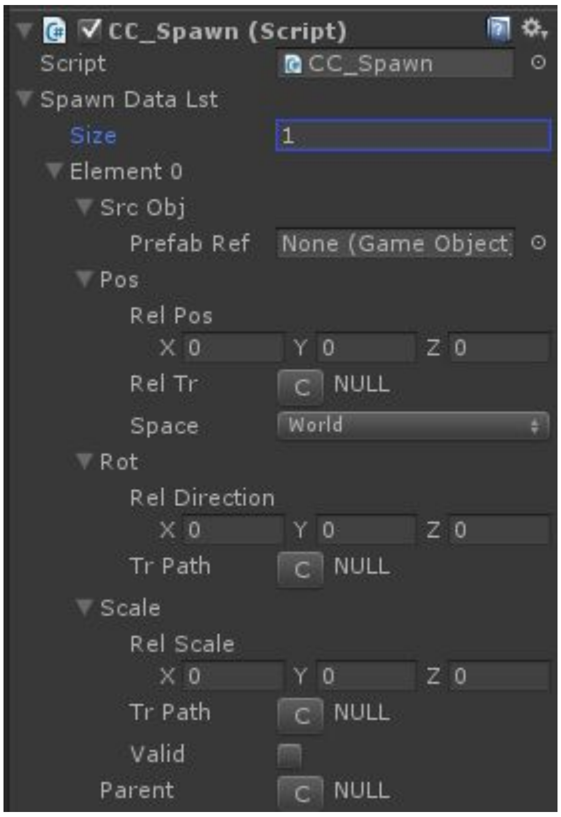
Duration: 0

Amplitude: 0

7) Spawn

Spawn component is used to create new GO during runtime.

In most cases, it could be replaced by other simpler components;


CC_Spawn (Script)

Script: CC_Spawn

Spawn Data Lst

Size: 1

Element 0

Src Obj

Prefab Ref: None (Game Object)

Pos

Rel Pos: X 0 Y 0 Z 0

Rel Tr: NULL

Space: World

Rot

Rel Direction: X 0 Y 0 Z 0

Tr Path: NULL

Scale

Rel Scale: X 0 Y 0 Z 0

Tr Path: NULL

Valid

Parent: NULL

SrcObj is used to specify an Prefab to be instantiated;

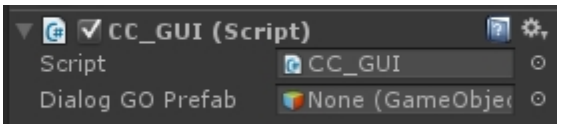
Pos is used to specify where the new GO is put.
RelPos is an offset position based on RelTr;
RelTr is used to specify the transform which the position is calculated based on.
If is null, means the CCRoot;
If is '/', means the Scene's Root;
If is a string 'X', means Transform path 'CCRoot/X'
Space specify how to calculate the world position, WORLD will add the RelTr's position and RelPos in world coordinate system, LOCAL will add the RelTr's position and RelPos in RelTr's coordinate system.

Rot is used to specify the new GO's rotation:
TrPath is used to specify a transform on which to calculate the direction. Could be NULL, which means calculate in world coordinate system.
RelDirection specify the direction of new GO's forward direction.

Scale is used to specify the new GO's scale:
The calculation method is same as Rot;

Parent is used to specify the parent transform for the new GO, if not specified, the GO has the Spawn component will be the parent of the new GO;

8) GUI
To use this component, you just need to select an GUI prefab from the project window, a new GO will automatically be created as child under this one, and you could tweak the parameters on the new GO.


CC_GUI (Script)
Script CC_GUI
Dialog GO Prefab None (GameObje

We provide several common built-in GUI prefabs for your referecne and prototyping usage. The built-in GUI prefabs are all implemented in UnityGUI and located at 'Assets/Skele/Res/GUITpl' directory.