Software Studio
軟體設計與實驗

Cocos Creator : UI

Hung-Kuo Chu
Department of Computer Science
National Tsing Hua University

CS2410

User Interface

- Send messages to game to tell it what you want to do.
- Show information to user.
- For example, show score, use button to call function…

UI Components

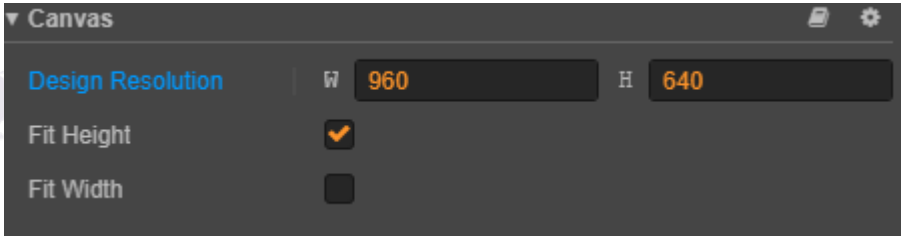
- Canvas
- Layout
- Button
- ScrollView
- EditBox
- Label
- RichText
- …

Canvas

- The **Canvas** component can get the actual resolution of the device screen and zoom in and out of all the rendered elements in the scene.
- There can only exist **one** Canvas component in the scene at a time.
- We recommend you set all the UI and renderable elements as Canvas‘s child nodes.

Canvas

- Design Resolution
  - the resolution blueprint used while the content producer builds the scene
- Fit Height
  - the height of the design resolution will auto-fit to the screen height
- Fit Width
  - the width of the design resolution will auto-fit to the screen width

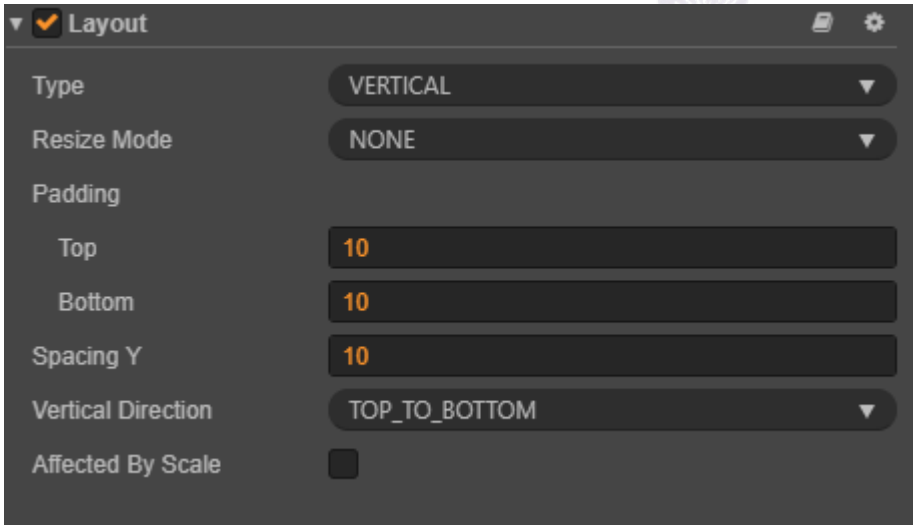


Layout

- Layout is a container component. It can unlock the auto-layout function to arrange all the sub-objects automatically, so that the user can use it to make a list, page turning and other functions conveniently.

Layout

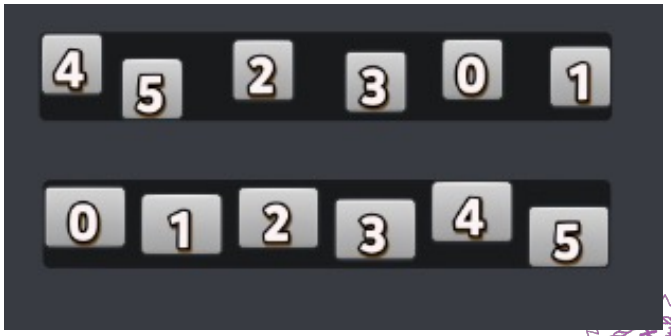
- Type
  - None
  - Horizontal
  - Vertical
  - Grid
- Resize Mode
  - None
  - Container
  - Children



Layout Example

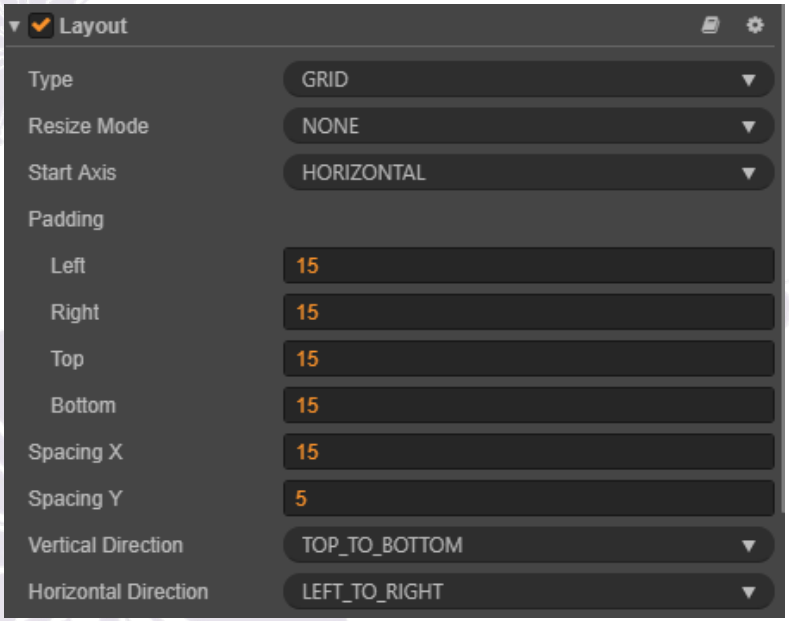
- Type: None
- Resize: Container

- Type: Horizontal
- Resize: Children

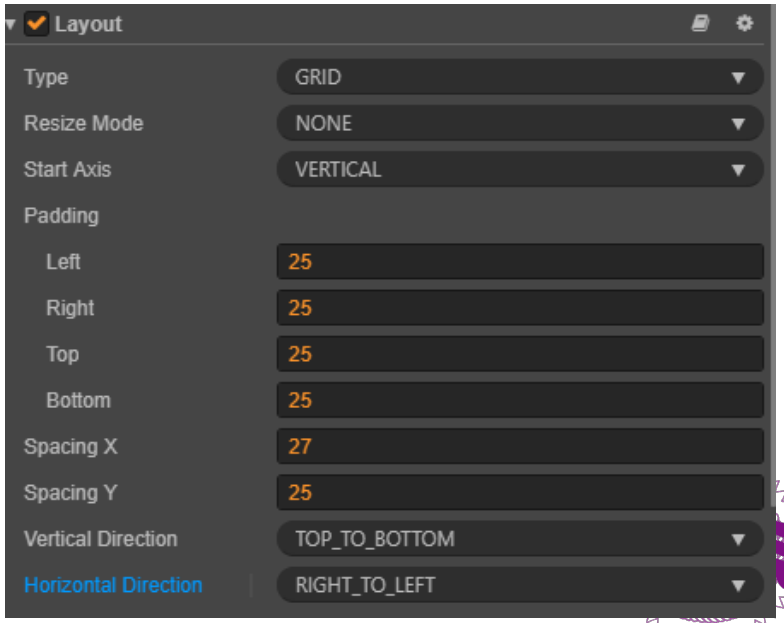


Layout Example

- Horizontal Grid



- Vertical Grid



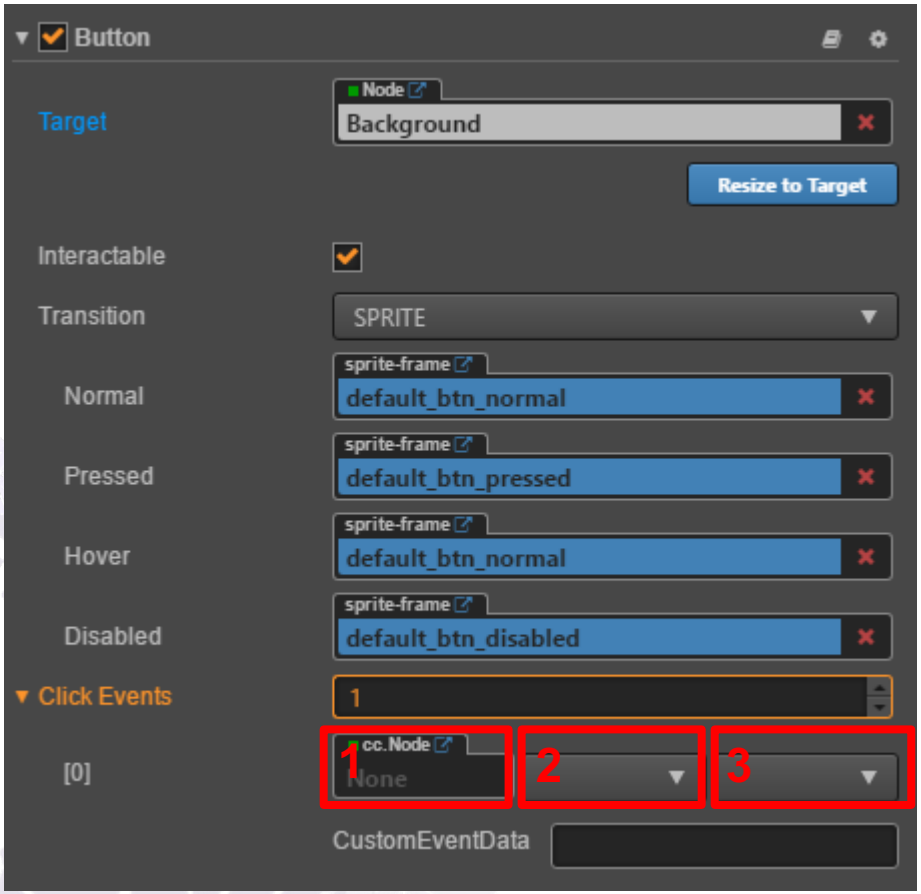
Button

- The button component responds to a click from the user.
- When the user clicks a Button, its status will change. In addition, users can assign a custom behavior to buttons' click event.

Button

- Transition : button appearance will change according to its state.
  - None
  - Color
  - Sprite
  - Scale
- Click Events : Functions will be called when user click the button.

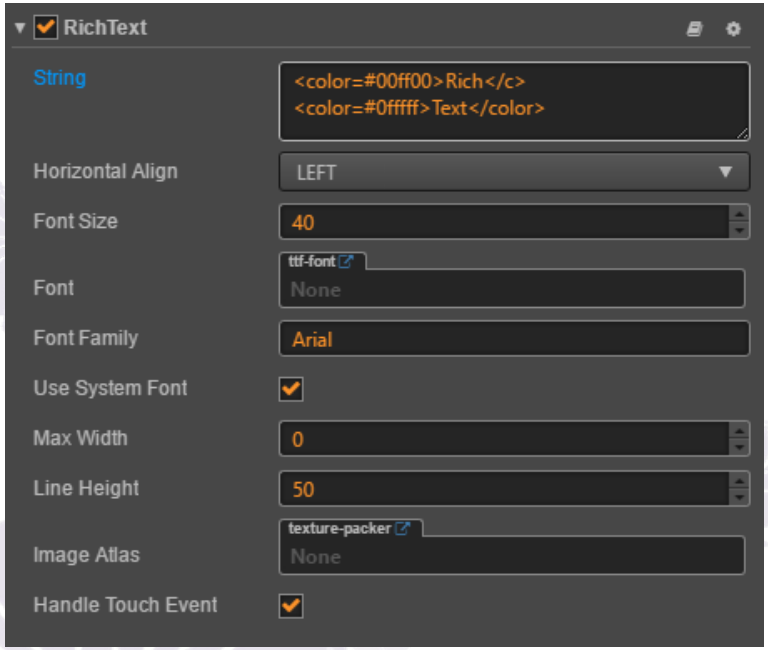
Button

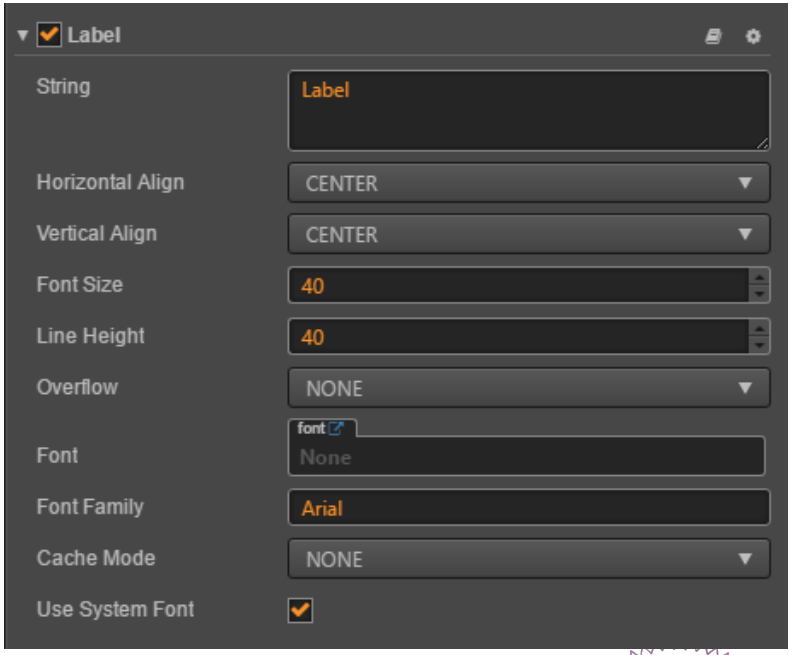


1. Script Node
2. Script Name
3. Function Name

RichText / Label

- These two components are used to show text with different properties.





RichText / Label

- String
  - The string we want to show in the scene.
- Align
  - Decide the way to align the text.
- Font
  - Use customized TTF font or font to adjust the style of string.

RichText Tags

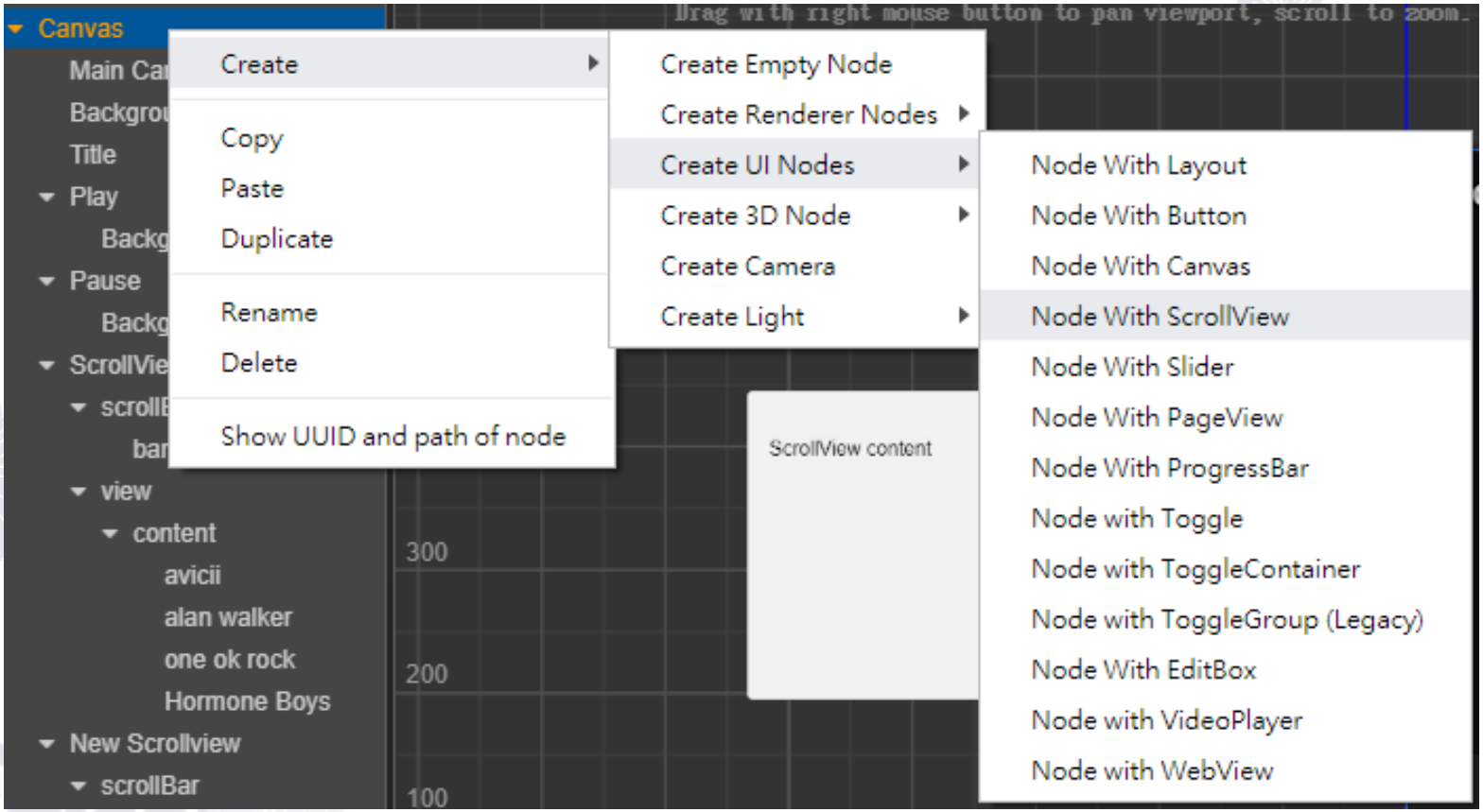
- RichText can use BBcode format to customize your string.
  - <color> : Specify the font rendering color
  - <outline> : Specify the font outline
    - you can customize the outline color and width by using the *color* and *width* attribute.
  - <b> : Bold font
  - <i> : Italic font
  - <u> : Add a underline to the text

<color=#00ff00>Cocos Creator</c> → Cocos Creator

ScrollView

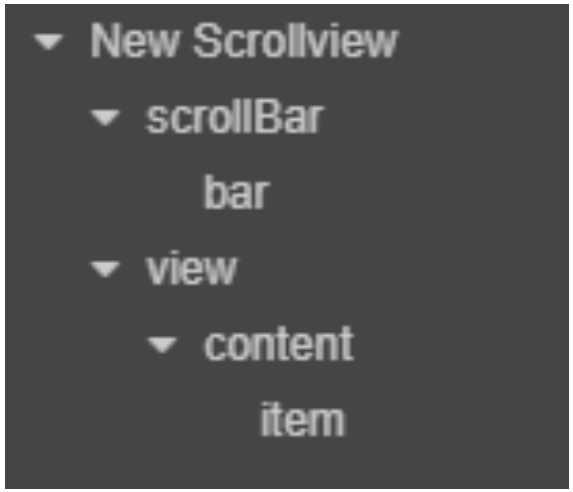
- It provides a way to browse more contents within a limited display area.
- Generally, ScrollView will be used along with the **Mask** component and the **ScrollBar** component can also be added to show the location of the browsing content.

ScrollView

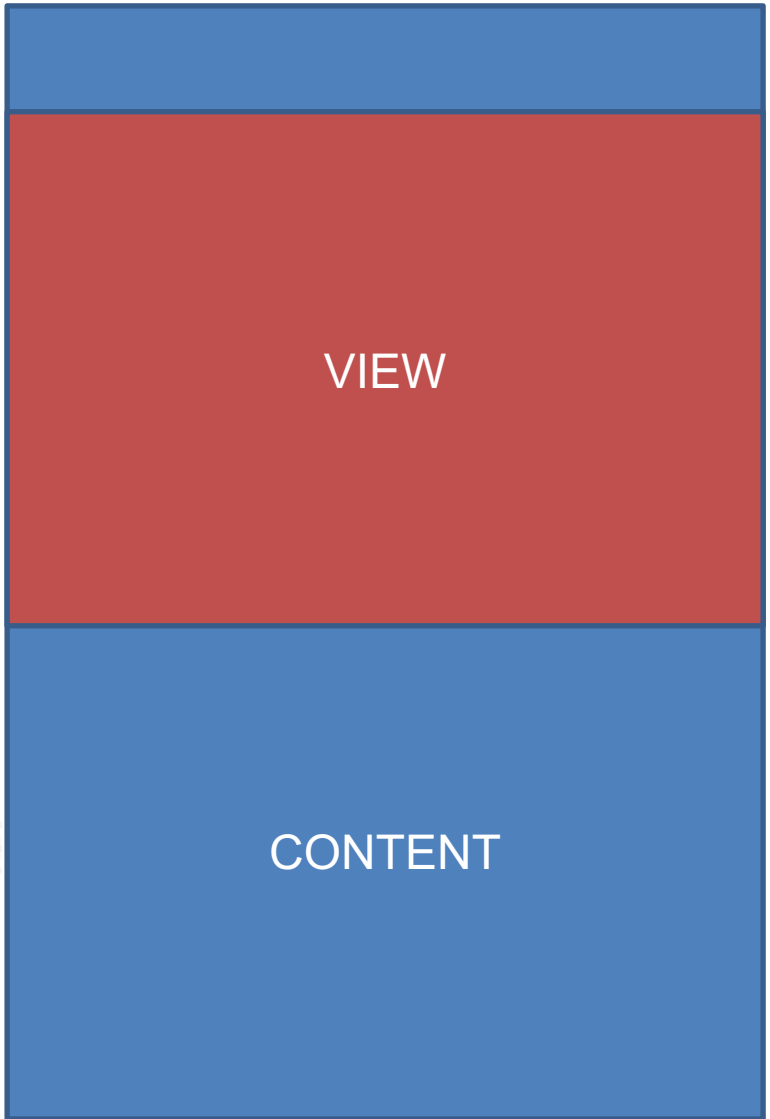

Canvas
Main Ca
Backgrou
Title
Play
Backg
Pause
Backg
ScrollVie
scrollB
bar
view
content
avicii
alan walker
one ok rock
Hormone Boys
New Scrollview
scrollBar
Drag with right mouse button to pan viewport, scroll to zoom.
Create
Copy
Paste
Duplicate
Rename
Delete
Show UUID and path of node
Create Empty Node
Create Renderer Nodes
Create UI Nodes
Create 3D Node
Create Camera
Create Light
Node With Layout
Node With Button
Node With Canvas
Node With ScrollView
Node With Slider
Node With PageView
Node With ProgressBar
Node with Toggle
Node with ToggleContainer
Node with ToggleGroup (Legacy)
Node With EditBox
Node with VideoPlayer
Node with WebView
ScrollView content
300
200
100

ScrollView

- Horizontal, Vertical
  - Set scrollview scrolling direction.
- Content
  - Scrollable area.
- View(Mask)
  - Limit the showing area.
- Brake
  - The deceleration coefficient after scrolling.



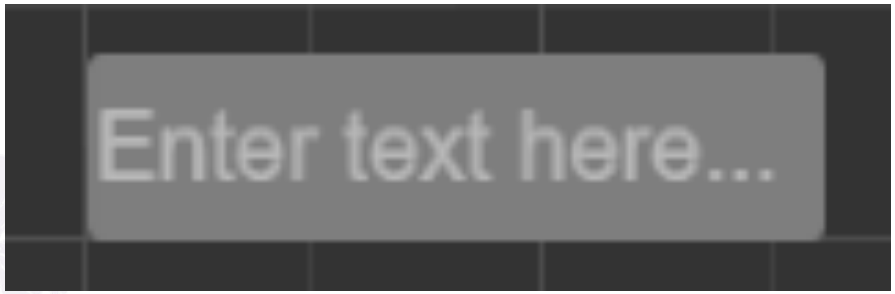
Content v.s. View



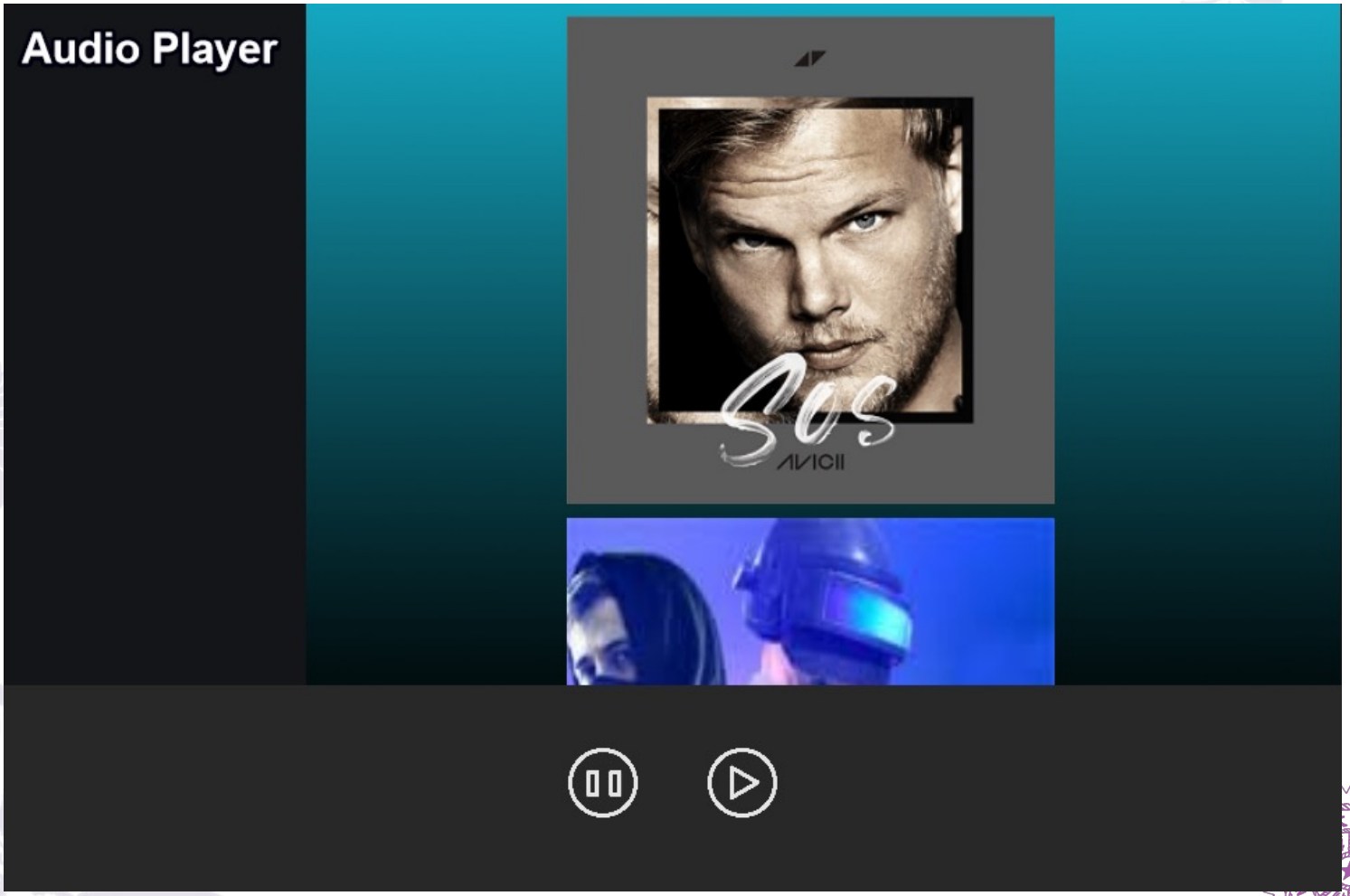
We can only see the content in the **view** area

EditBox

- EditBox is a **text input** component, you could use this component to get user input easily.

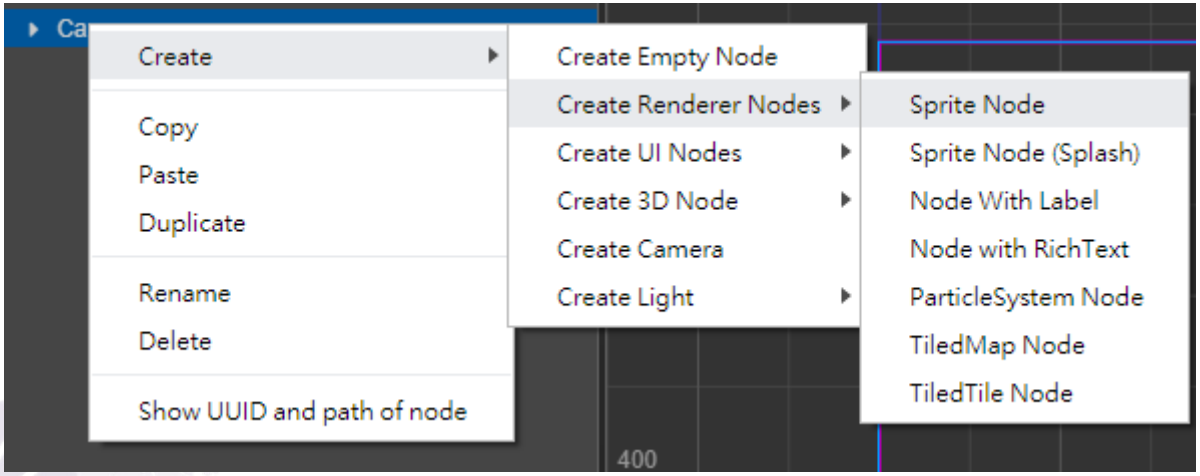


Simple Music Player



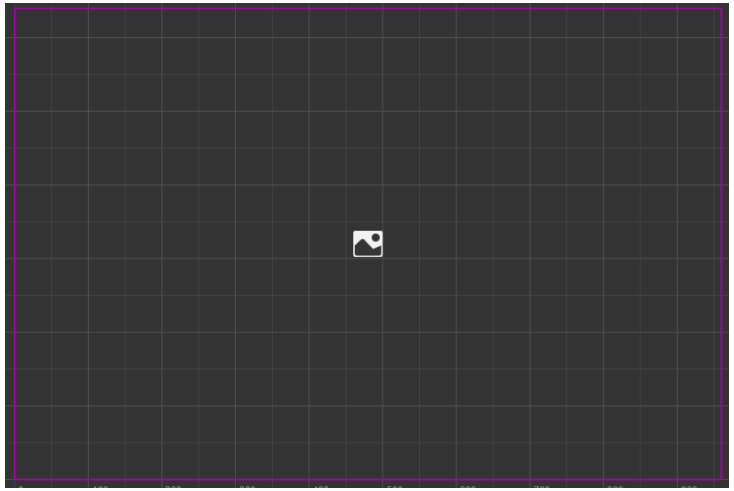
Create Sprite

- Right-click on the Canvas node of Node Tree [Create] -> [Create Renderer Nodes] -> [Sprite Node]



Let the Sprite Fill the Canvas

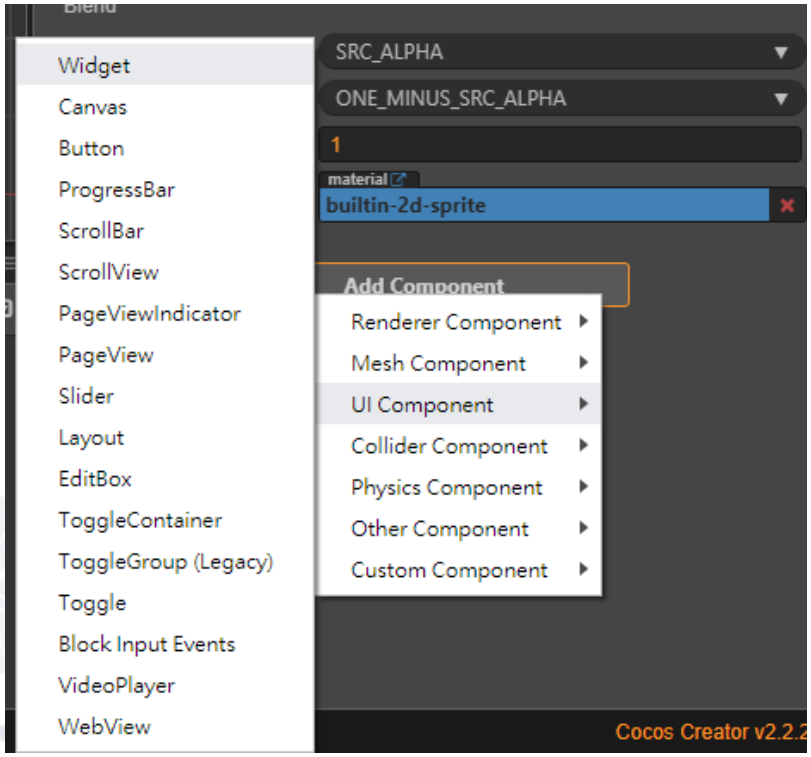
- Sprite node starts with a small picture.



- We want the Sprite node to use our image as source and fill the canvas

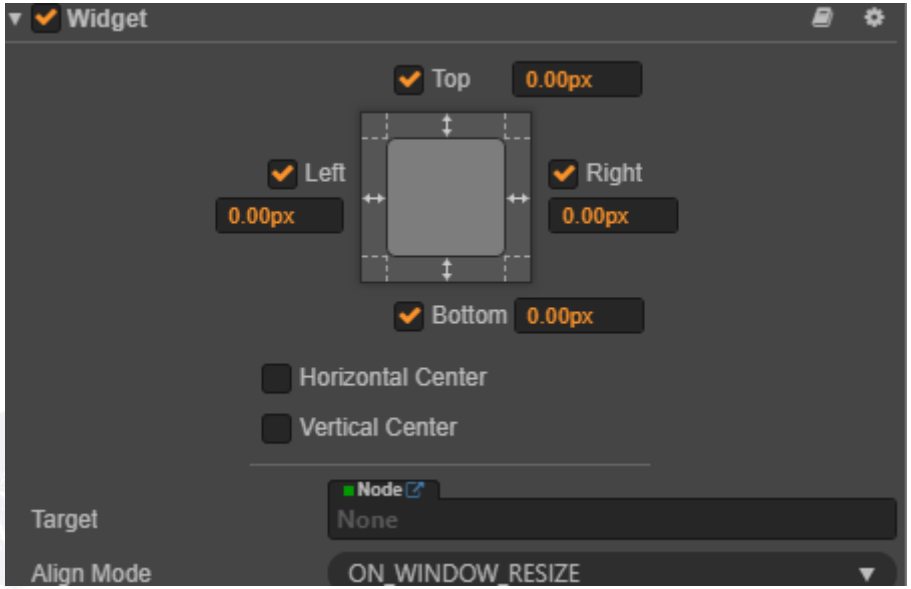
Widget

- Add a Widget component to the Sprite node by [Add Component] -> [UI Component] -> [Widget]



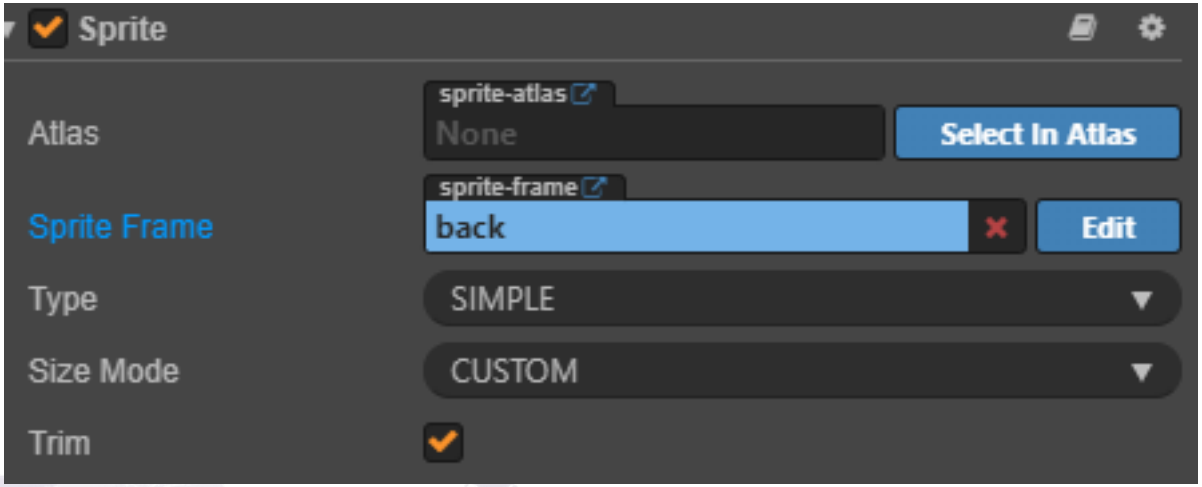
Widget

- Check Top, Bottom, Right, Left, and set the value to 0. Then the Sprite will match the parent node's size.



Replace Sprite Frame

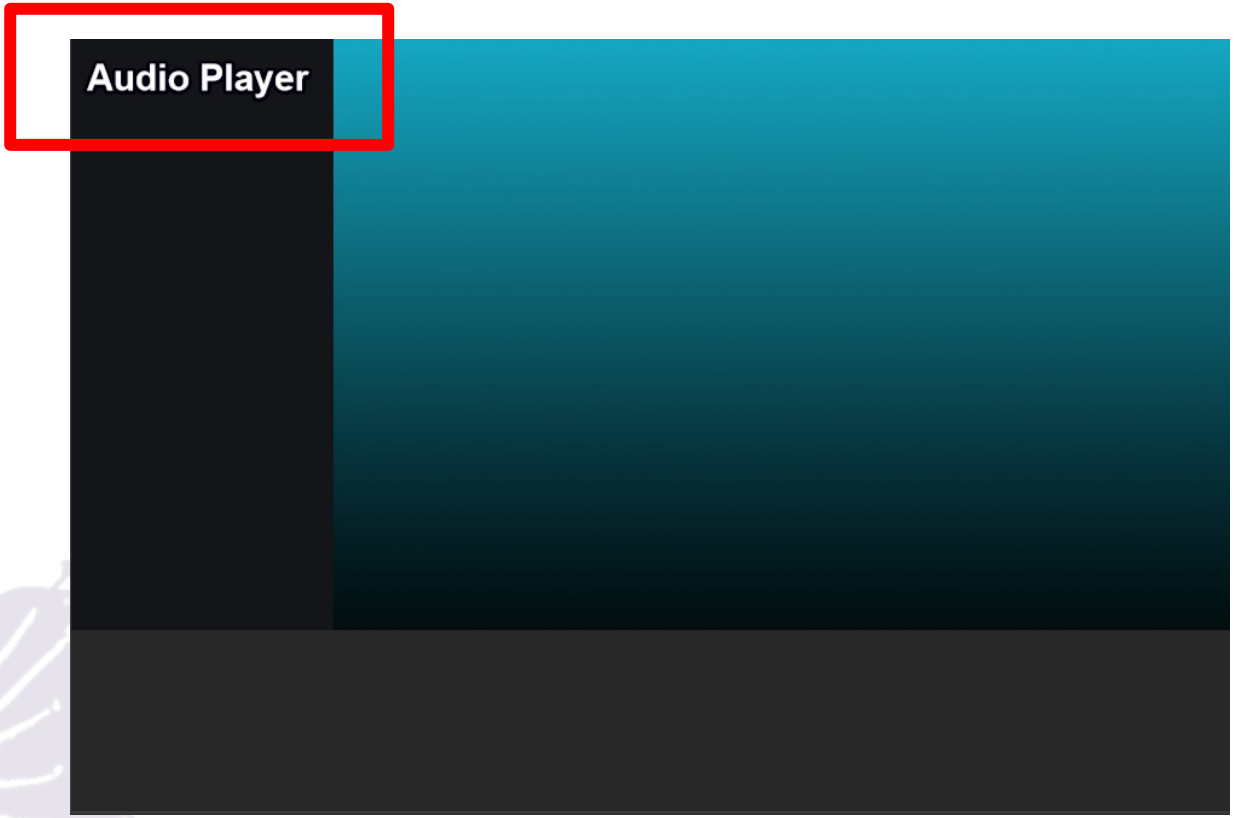
- Replace the Sprite Frame with the image we want to use as the background



Background Image

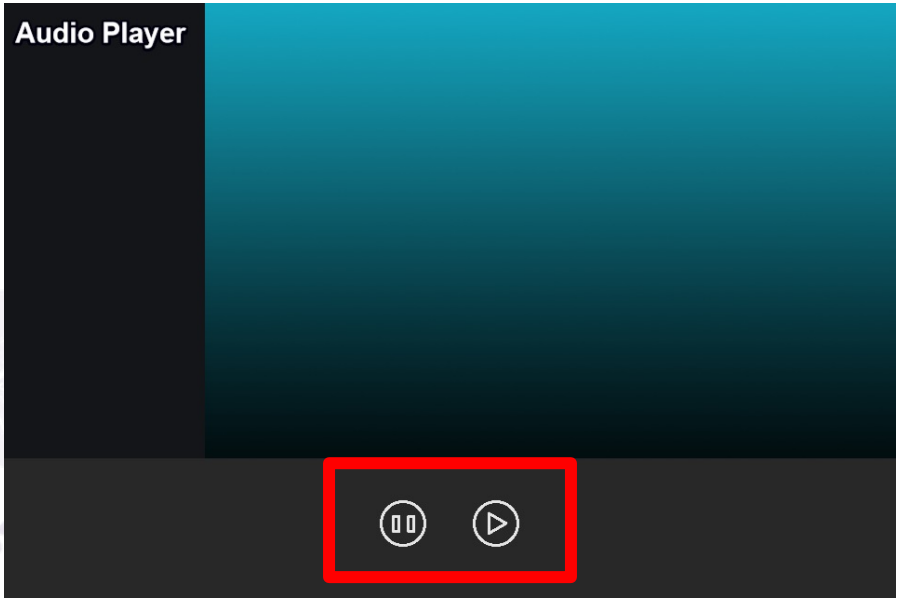
Add Rich Text

- [Create] -> [Create Renderer Nodes] -> [Node with RichText]



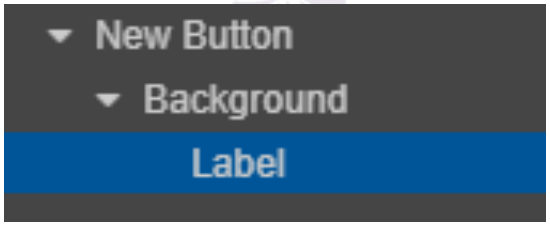
Button

- Then, we add two buttons to let user play or pause the music. [Create] -> [Create UI Nodes] -> [Node with Button]
- Add ClickEvents to the button so that we can call our function by these buttons.



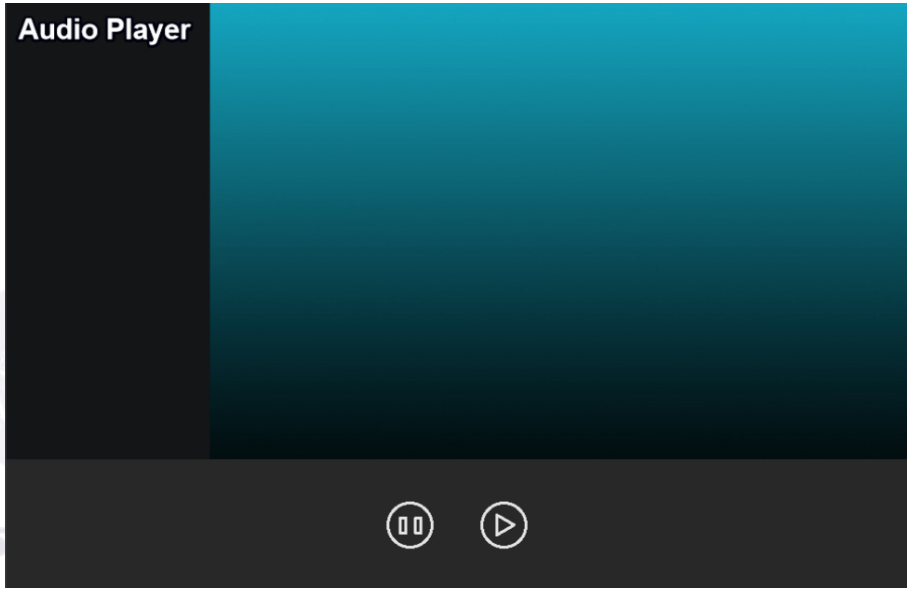
Change Button Style

- The newly added button node will have two child nodes of background and label by default
  - Change the interaction with the button → New Button
  - Change the background image → Background
  - Change text content → Label
    - You can delete the label node directly if you don't want the text



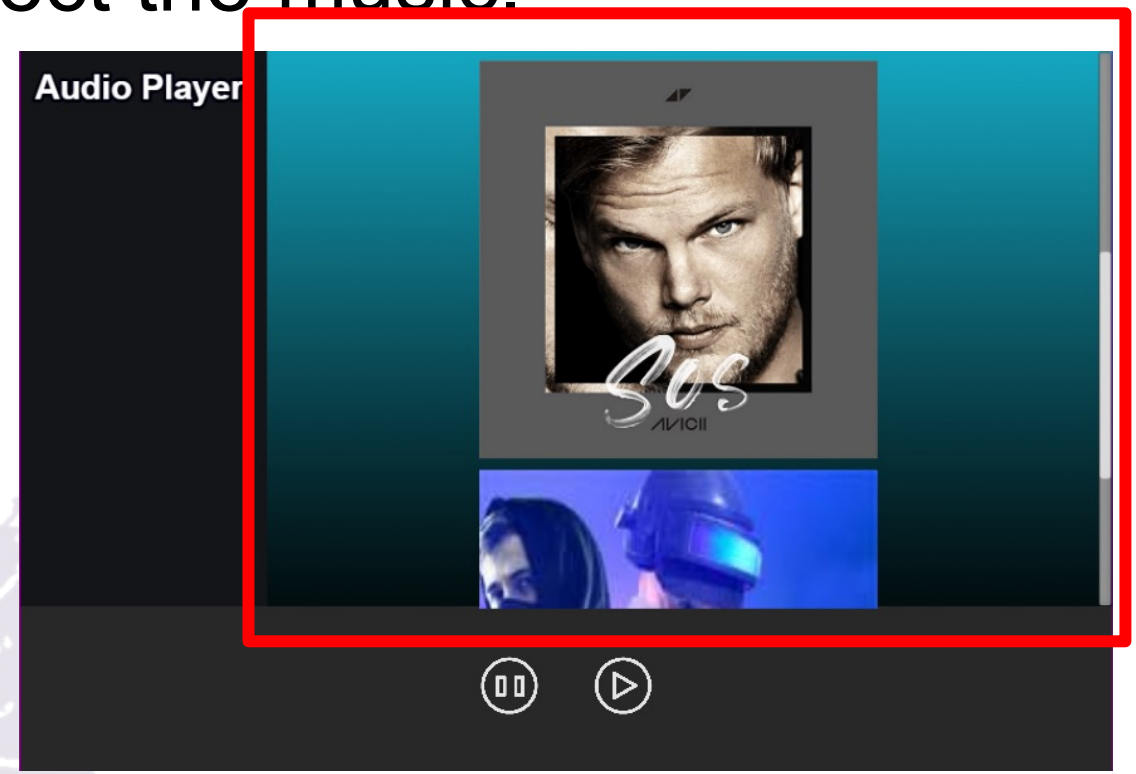
Change Button Style

- To make our buttons more beautiful, we use other sprites as buttons' background.
- Change one button's transition to *color* and another to *scale*.



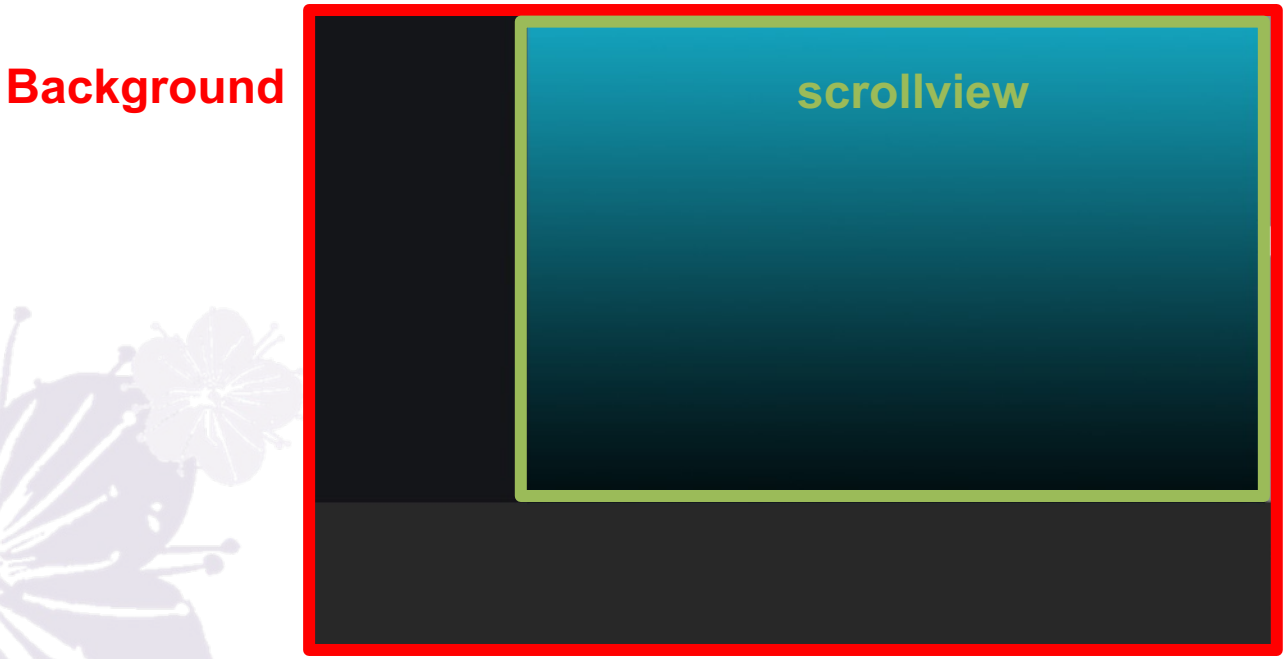
Song Selector

- Here we use **scroll view**, **layout** and **button** to create a small window for user to select the music.



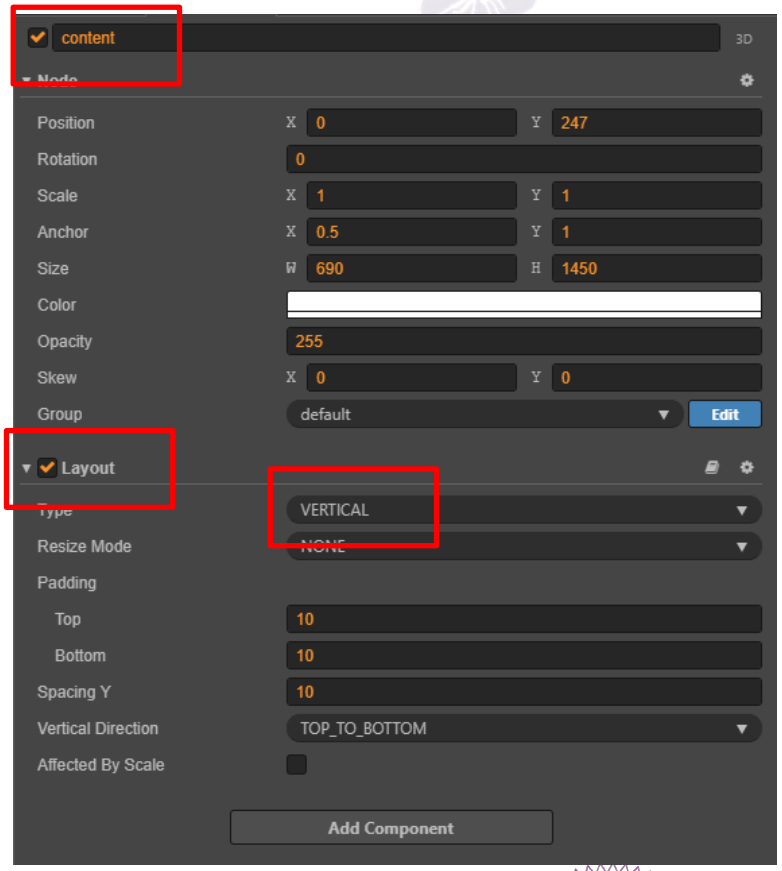
Song Selector (Cont'd)

- Create new a scrollview node
- Set the scrollview size to x:742.7 & y:490
- Drag the scollview node to the correspond position



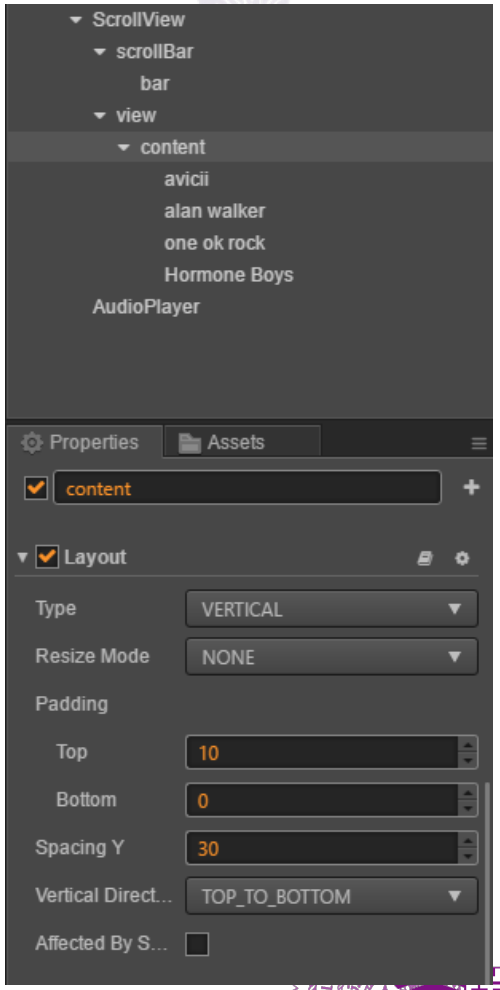
Song Selector (Cont'd)

- Scroll view and layout let us show more nodes in a limited area easily.
- Add **Layout** component to scroll view's **content** and set it's type as **Vertical**.



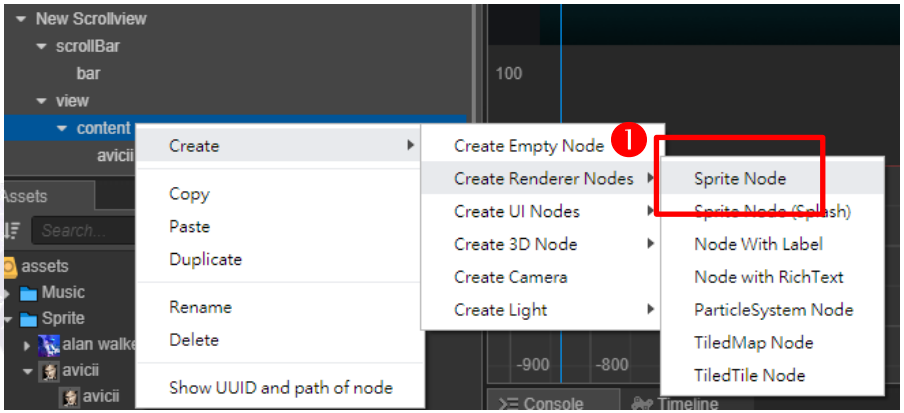
Song Selector (Cont'd)

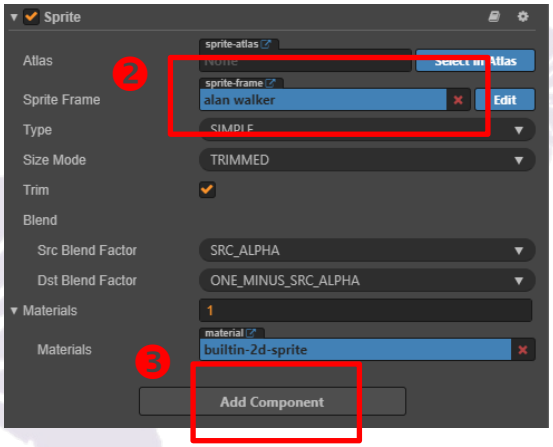
- We can modify **SpacingY** to control the interval of each node in this layout.
- Then we can add our sprites into **content** and you will see them arranged vertically.

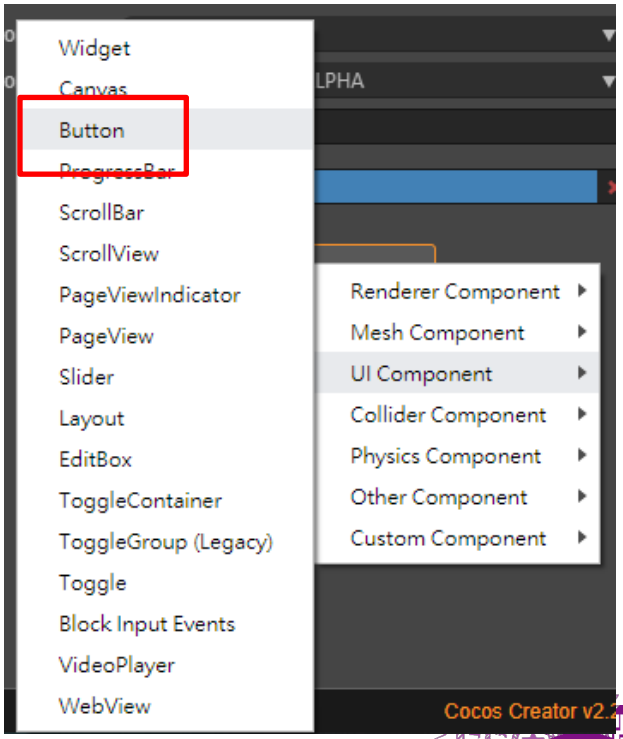


Song Selector (Cont'd)

- Add **Button** component which we will use to trigger script to our sprites and play the chosen music.







BGM and SE

- Most of the games have background music (BGM) or sound effects (SE).
- We can also import audio files in Cocos Creator and play them during game execution
- Cocos Creator can play sound in two ways
  - **AudioSource**
  - **AudioEngine**

Play Audio in Cocos

- Both **AudioEngine** and **AudioSource** can play audio.
- The difference between them is that **AudioSource** is a component that can be added to the scene and set by the editor. **AudioEngine** is a pure API provided by the engine and can only be called in scripts

Play Audio in Cocos

- It is currently recommended to use the **audioEngine.play** interface to play audio uniformly.
- Or you can use two interfaces, **audioEngine.playEffect** and **audioEngine.playMusic**, the former is mainly used to play sound effects, and the latter is mainly used to play background music. See API documentation for details.

AudioEngine

- If we want to use AudioEngine to play background music, there are several common methods
  - cc.audioEngine.playMusic(clip, loop)
  - cc.audioEngine.resumeMusic();
  - cc.audioEngine.pauseMusic();

AudioPlayerController

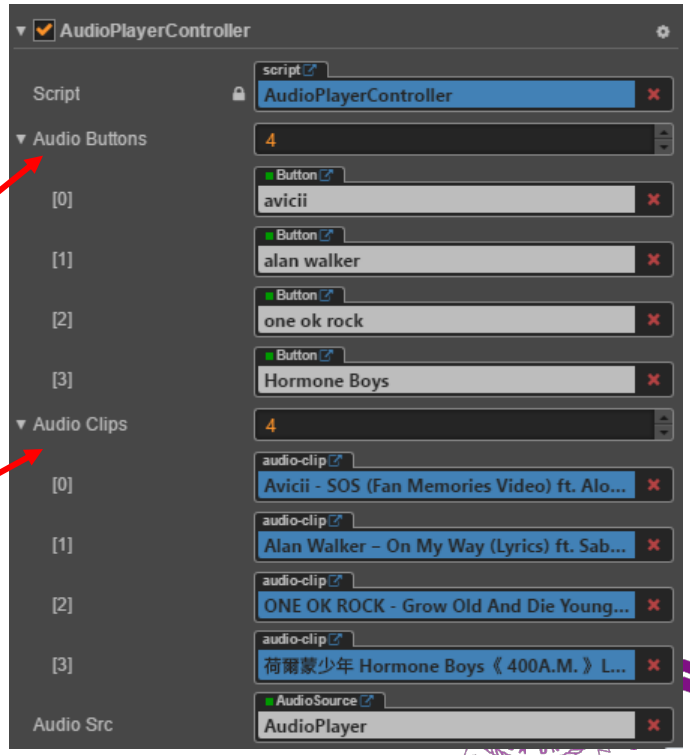
- In our player, we want to play not only one song.
- Hence, we use **Script** to create two arrays to load our music files and their buttons, and mount script on them later.

AudioPlayerController.ts

```
@property([cc.Button])
audioButtons: cc.Button[] = [];

@property({type:cc.AudioClip})
audioClips: cc.AudioClip[] = [];;
```

Canvas/AudioPlayerController



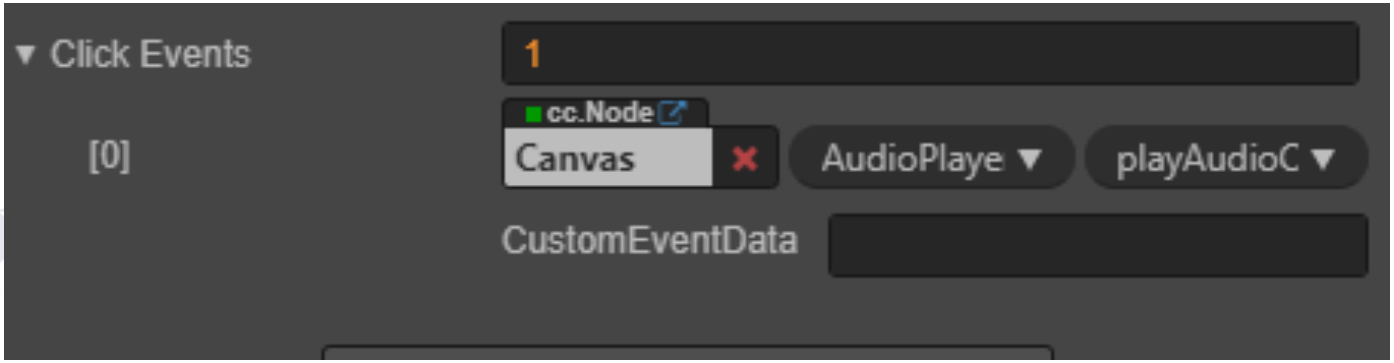
AudioPlayerController

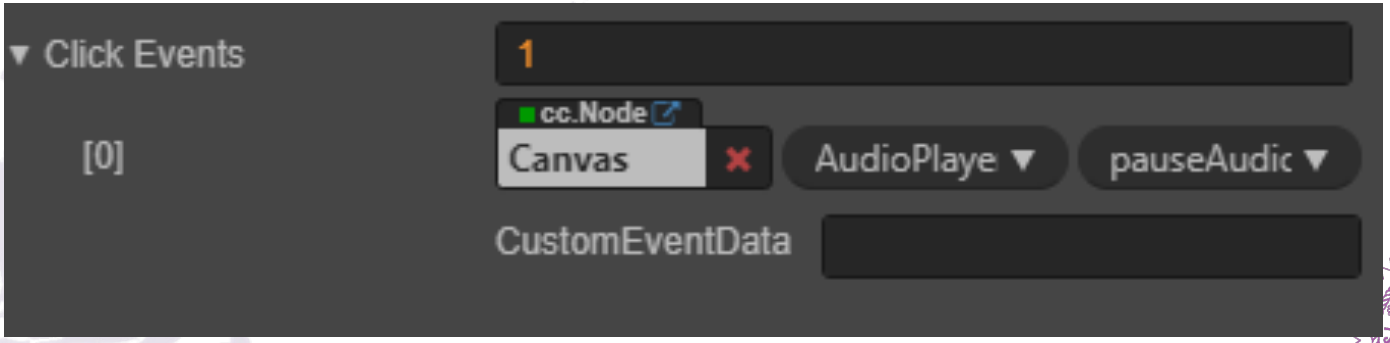
- Create two functions to control music pause and resume.

```
playAudioClip(){
    cc.audioEngine.resumeMusic();
}
pauseAudioClip(){
    cc.audioEngine.pauseMusic();
}
```

AudioPlayerController

- Bind the functions to the click event of the pause and resume buttons





AudioPlayerController

- Using the dynamic event binding to add clickevent to sprites in the song selector

```
start () {
    for(var i=0; i<this.audioClips.length; i++){
        var clickEventHandler = new cc.Component.EventHandler();
        clickEventHandler.target = this.node;
        clickEventHandler.component = "AudioPlayerController";
        clickEventHandler.handler = "changeAudioClip";
        clickEventHandler.customEventData = "" + i;
        this.audioButtons[i].clickEvents.push(clickEventHandler);
    }
}
```

```
changeAudioClip(event, customEventData){
    cc.audioEngine.playMusic(this.audioClips[customEventData], false);
}
```


thank you!
Question?