How To Create CAPH Application Using Scene

Published 2014-10-28 | (Compatible with SDK 5.1 and 2014 models)

> Tutorial on make caph application use Scene.

Contents

Overview

A scene is a graphical concept providing a screen with which you can interact. It draws its user interface with a bunch of widgets. So, in widget's hierarchy, it becomes a root object placed just underneath a UI context.

An application usually consists of several scenes that are loosely coupled to each other, and a scene can call another scene to be shown. When two scenes are in transition, you need to destroy widgets bound to the old scene and create widgets for the new scene. You also need to change the current focus to a widget on the new scene using HighlightHelper and to get KeyControl to update its navigation map which helps finding a next widget on a certain user's interaction such as left, right, up, or down key event.

SceneManager provides a mechanism to help you to concentrate on compositing your scene without considering such things which you repetitively need to handle. It handles scene's lifecycle. When scenes are in transition, it notifies each scene what the scene needs to do. There are several callback methods in a scene and those are invoked when the scene initialization, creation, rendering, or destroy is needed. Each callback provides you the opportunity to perform specific work that's appropriate to the changes.

To create a scene, you need to make a subclass inherited from Scene and register it to SceneManager. Once the scene is registered and the callback methods in the subclass are implemented by user, then SceneManager can properly manage the transition between scenes.

Prerequisites

To create a CAPH Application using scene, for example, you should include caph.wui dependencies, basic widget of UI, the base style of UI, and app source files in your application by putting the following code in the <head> and <body> sections of index.html, as follows:

```
<html>
  <head>
    <script src="$CAPH/1.0.0/caph-level1-unified.min.js"></script>
    <link href="$CAPH/1.0.0/caph.css" rel="stylesheet" type="text/stylesheet">

    <script src="myscenes.js"></script>
    <script>
      var flag = false;
      function func_onLoad() {
        if(typeof curWidght != 'undefined') {
          curWidget.setPreference("ready", "true");
        }
        if(!caph.platform.dtv.Browser || caph.platform.dtv.Browser.browserType().sectvbd) {
        } else {
          caph.app.run();
        }
      }
      window.onShow = function (e) {
        if(flag === false) {
          flag = true;
          caph.app.run();
        }
      }
    </script>
  </head>
  <body onload="func_onLoad();" style="background-color : #000000">
  </body>
</html>
```

Source files mentioned above are explained in the following table.

File	Description
caph-level1-unified.min.js	It includes all kinds of basic widgets for UI which will be used. It also includes external libraries
caph.css	The class style of caph.wui.
myscenes.js	File created by you to includes widgets in a scene.

Environment

In order to use Samsung Smart TV SDK and caph.wui to run your applications on TV screen, you might need a text editor to create files that comprise HTML, JavaScript and CSS files for your applications. Samsung Smart TV is required to verify if whether your application is running well or not. You can also use the emulator provided with the SDK to debug and test the applications before uploading them on to the TV device.

Source Files

> Note
>
> The files needed for the sample application are "index.html" and "myscenes.js".
>
> To download source files, click **here**.

How to Develop

What you need to do next is to create a scene with widgets you want and incorporate it with SceneManager. Open your JavaScript file, “myscenes.js”, and get an instance of SceneManager and call its addSceneEventHandler() function to register your scene which takes two parameters where first parameter is unique identifier for your scene, it MUST be unique in your application and second parameter is a scene object which will be a subclass of Scene. Your scene object SHOULD

implement the onCreate() method, which SceneManager will call when the scene needs to be created. Inside the onCreate(), you can create widgets you want. When you are done creating widgets in it, you don't need to call the render() method separately as a final step of widget creation, unlike the way to create a widget outside of scene. You can use addWidget() method instead of calling each widgets' render function. Additionally, if you change Scene in SceneManager, Scene is initialized on its own.

> Note
>
> There are more callback methods for its subclass in Scene. For more information about Scene, see the Scene guide.

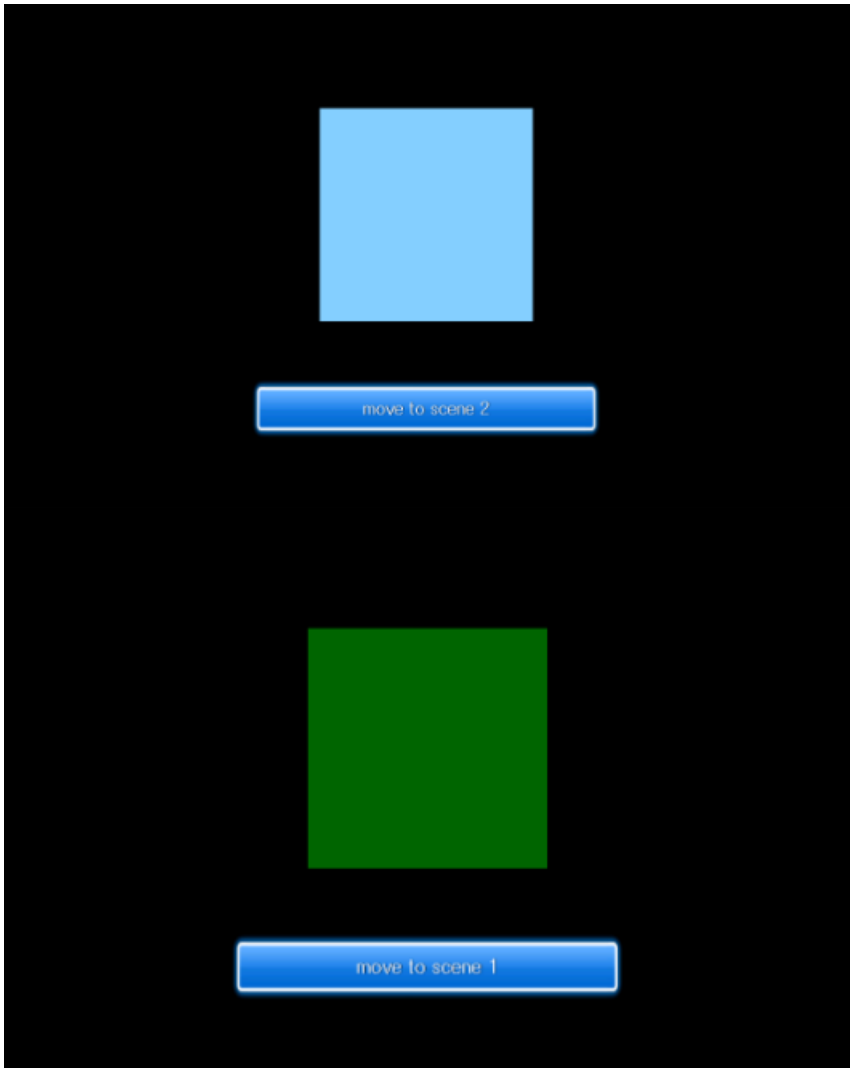
```
1  (function() {
2    // Set aliases for Box, Button and so on.
3    var Box = caph.wui.widget.Box;
4    var Button = caph.wui.widght.Button;
5    var scenemanager = caph.wui.widget.SceneManagergetInstance();
6
7    // create a scene named 'scene1'
8    scenemanager.addSceneEventHandler('scene1', {
9      onCreate : function(context) {
10       this.super(context);
11       // Build a Button
12       var button = new Button( {width:500} );
13       button.setText('move to scene 2');
14       button.setAbsolutePosition('50%', '80%', 2);
15       button.addEventListener(click, function() {
16         // scene1 is initialized
17         scenemanager.showScene('scene2');
18       });
19       // Build a Box
20       var box = new Box( {width:300, height:300} );
21       box.setStyle('background', 'lightskyblue');
22       box.setAbsolutePosition('50%', '50%', 1);
23
24       this.addWidget(box);
25       this.addWidget(button);
26     }
27   });
28
29   // create a scene named 'scene2'
30   scenemanager.addSceneEventHandler('scene2', {
31     onCreate : function(context) {
32       this.super(context);
33       // Build a Button
34       var button = new Button( {width:500} );
35       button.setText('move to scene 2');
36       button.setAbsolutePosition('50%', '80%', 2);
37       button.addEventListener(click, function() {
38         // scene1 is initialized
39         scenemanager.showScene('scene1');
40       });
41       // Build a Box
42       var box = new Box( {width:300, height:300} );
43       box.setStyle('background', 'darkgreen');
44       box.setAbsolutePosition('50%', '50%', 1);
45
46       this.addWidget(box);
47       this.addWidget(button);
48     }
49   });
50 }());
```

The description of the example code is :

Line 8	create a scene named 'scene1' and register it to SceneManager.
Line 9	implement a function callback for scene creation.
Line 12-18s	create a button for the scene and add an event listener to respond the button's on-click event to go to the scene defined 'scene2'.

Line 20-22s	create a box to differentiate the scene.
Line 24-25s	add above two widgets to the scene.
Line 30	create a scene named 'scene2'.
Line 31	implement a function callback for scene creation.
Line 34-40s	create a button for the scene and add an event listener to respond the button's on-click event to go to the scene defined 'scene1'.
Line 42-44s	create a box to differentiate the scene.
Line 46-47s	add above two widgets to the scene.

The screenshots of this example look like this :



Related Document

- How to customize CAPH WUI Widgets
- How to create CAPH Application
- How to Integrate CAPH WUI Widgets with jQuery Library
- How to Use Animation with CAPH WUI Widgets
- How to Use CAPH WUI Widgets with HTML
- How to use CAPH WUI Widgets
- CAPH Framework Overview