Transitioning to SceneGraph

The following summarizes how to transition from the older Roku API to SceneGraph applications. In general, SceneGraph handles events without requiring you to write custom event loops, and allows you easily add custom event handlers by observing virtually any of the fields used in your SceneGraph application. Also, SceneGraph supplies built-in node classes that are very similar to some BrightScript UI screen components, but also gives you much greater flexibility to build your own custom screens. Because SceneGraph is more dynamic, flexible, and object-oriented than the older Roku API for building UI elements, you cannot write an equivalent SceneGraph application by simply substituting different function names and argument lists for UI elements.

However, much of your existing BrightScript code that handles other aspects of your application can probably be used in the new context of a SceneGraph application. For example, calls to your existing BrightScript functions can be added to any SceneGraph XML component file, by specifying the location of the BrightScript file in the package components directory in the **<script>** element. For example, if you had a set of functions to manipulate URLs, you could add the following **<script>** element to allow a SceneGraph component to call the functions:

```
<script type = "text/brightscript" uri = "pkg:/components/uriutils.brs" />
```

If these calls relied on application global data, you would have to set up a *global node* as described in **Global Scope** for this data.

There is a simplifying assumption you can make when transitioning an older Roku API application to SceneGraph. In many cases, SceneGraph replaces a series of BrightScript component interface function calls with setting and reading of node class fields of the equivalent parameters. For example, the SetNumPinEntryFields() interface function call for the roPinEntryDialog component is replaced by setting the `pinLength` field of the internal **PinPad** node object of a **PinDialog** node. But note that for many of the older Roku API screen components, you must develop your own custom screen component equivalent from more fundamental SceneGraph node classes. For example, you must develop an equivalent of the roSearchScreen component using the **MiniKeyboard** node class as the user input basis of a custom SceneGraph search screen component. You must then write additional BrightScript code to replicate the function of the roSearchScreen component.

BrightScript Component	SceneGraph Equivalent	Implementation Notes
roUniversalControlEvent	**onKeyEvent()**	SceneGraph supplies a single function that replaces the need to write a remote control input event handling loop.
roTimespan	**Timer** node	Adding a **Timer** node, observing the `fire` field, and writing an event handler that scripts some aspect of another node is the equivalent of writing a loop including roTimespan.
BrightScript Component Events	SceneGraph events	You can observe changes in virtually all aspects of SceneGraph operation using the **<interface>** field `onChange` attribute event, or adding an observer to a field of a SceneGraph node.
roPinEntryDialog	**PinDialog** node	The function of the **PinDialog** node is almost identical to the roPinEntryDialog component.
roTextScreen	**ScrollableText** node	These two are very similar in appearance and operation.
roFont, roFontRegistry	**Font** node	The **Font** node simplifies specifying different fonts.
roGridScreen	**PosterGrid**, **MarkupGrid**, **RowList** nodes	The built-in SceneGraph poster grid node classes are simpler, but much more flexible. You can use SceneGraph to create a custom grid screen identical to the older Roku API, but even better, you can create a completely original grid screen of your own design.
roImageCanvas, roScreen	SceneGraph	SceneGraph starts with the ability to draw your own custom screen elements, then use them as objects throughout your application.
roKeyboardScreen	**MiniKeyboard**, **Keyboard** nodes	The keyboard screens and SceneGraph keyboard node classes are very similar in appearance and operation.
roListScreen	**LabelList**, **MarkupList**, **RowList**, **CheckList**, **RadioButtonList** nodes	SceneGraph gives you many more options and flexibility for list screens.

roAudioPlayer	**Audio** node	The roSpringboardScreen component for user selection of audio files can be created using SceneGraph list and grid node classes.
roVideoScreen, roVideoPlayer	**Video** node	The SceneGraph **Video** node class encompasses all the functionality of the roVideoScreen and roVideoPlayer components. The roSpringboardScreen component for user selection of video files can be created using SceneGraph list and grid node classes.
roSearchScreen	**MiniKeyboard** node	The **MiniKeyboard** node class can be used as the user input basis of a custom SceneGraph component equivalent to the roSearchScreen component.
roSpringboardScreen	SceneGraph	The roSpringboardScreen component for user selection of audio/video files can be created using SceneGraph list and grid node classes.