Developing SceneGraph Applications

Table of Contents

Channel packages should include all the XML components they define in a top level directory named `components` (the top level directory in the channel application package contains the `manifest` file, the `source` directory, and so forth). When the channel is launched all the files with extension `.xml` in the `components` directory are loaded and added to the available types of nodes that can be created.

Currently, channel packages must include all of the XML component XML files that they use. In the future, we may provide a library of useful XML components for developers as well.

Set Up the Application Directory

Create an application directory with the following minimum subdirectories and files:

- `manifest` file
- `source` directory
- `components` directory
- `images` directory

Name this directory for the application it will contain, such as VideoStore. In most cases you will probably just copy an existing application directory (such as provided for the **SceneGraph XML Tutorial**), and modify, add, and rename files and directories for your new application, as described below.

Create or Modify a manifest File

The `manifest` file contains the following fields:

```
title=application_title
subtitle=application_subtitle
major_version=major_version_number
minor_version=minor_version_number
build_version=build_version_number

mm_icon_focus_fhd=FHD_focus_graphic_file_URI
mm_icon_focus_hd=HD_focus_graphic_file_URI
mm_icon_side_hd=HD_side_graphic_file_URI
mm_icon_focus_sd=SD_focus_graphic_file_URI
mm_icon_side_sd=SD_side_graphic_file_URI

splash_screen_sd=SD_splash_screen_graphic_file_URI
splash_screen_hd=HD_splash_screen_graphic_file_URI
splash_screen_fhd=HD_splash_screen_graphic_file_URI
splash_color=color_specifier
splash_min_time=milliseconds
```

Many of the fields of the `manifest` file are optional or not required for the application to run. The `title` and `subtitle` fields can be useful for

debugging, as are the `_version` fields. These can be set to aid debugging various versions of your application, but are not required.

The various *_graphic_file_URI* field values are used to specify a graphic that allows a user to select your application, and a "splash" screen that appears briefly before the application begins to run, often used while it is still loading.

The four application selection graphic files consist of two files each for both HD and SD user displays, "focus" and "side". These two files are specified in a SceneGraph `manifest` file only for backwards compatibility with past application selection user interfaces. The current SceneGraph application selection user interface only uses the focus graphic file.

The two application splash screen graphic files are for SD and HD user displays. Splash screens are not required but are recommended for your application. The `color_specifier` field value sets a default or opening display screen color before the splash screen loads, or if splash screen graphic files are not specified. The `splash_min_time` field sets the time in milliseconds that the splash screen should be displayed regardless of the load status of the application.

The graphics files specified in the manifest file should be included in the application package `images` directory, so the URI to set the path to the files should use `pkg:` resource prefix, such as `pkg:/images/splash_sd.jpg`. More information about the requirements and recommendations for the graphics files included in the application package can be found in **Set Up the images Directory**.

The following is an example of a `manifest` file.

manifest File Example

```
title=Test Application
subtitle=A SceneGraph Test
major_version=1
minor_version=1
build_version=00001

mm_icon_focus_hd=pkg:/images/MainMenu_Icon_Center_HD.png
mm_icon_side_hd=pkg:/images/MainMenu_Icon_Side_HD.png
mm_icon_focus_sd=pkg:/images/MainMenu_Icon_Center_SD43.png
mm_icon_side_sd=pkg:/images/MainMenu_Icon_Side_SD43.png

splash_screen_sd=pkg:/images/splash_sd.jpg
splash_screen_hd=pkg:/images/splash_hd.jpg
splash_color=#000000
splash_min_time=1000
```

Set Up the source Directory

The source directory contains BrightScript code for the main applications execution thread, with the extension `.brs`. Since applications including SceneGraph scenes allow BrightScript code to either be embedded in, or used by, an XML component file in a **<script>** element, for many SceneGraph applications this directory will only contain a `main.brs` file to start the application. The `main.brs` file only need have enough BrightScript code to start the application by creating and displaying the scene specified in the SceneGraph scene.

The following shows a `main.brs` file that starts the application by creating and showing the scene defined in the SceneGraph scene named `rectangleScene` (the `rectangleScene` scene is defined in an XML component file in the `components` directory as described in **Set Up the components Directory**).

Example main.brs File

```
sub Main()
  showChannelSGScreen()
end sub

sub showChannelSGScreen()
  print "in showChannelSGScreen"
  screen = CreateObject("roSGScreen")
  m.port = CreateObject("roMessagePort")
  screen.setMessagePort(m.port)
  scene = screen.CreateScene("rectangleScene")
  screen.show()

  while(true)
    msg = wait(0, m.port)
    msgType = type(msg)

    if msgType = "roSGScreenEvent"
    if msg.isScreenClosed() then return
    end if
  end while
end sub
```

You can use this example `main.brs` file for your SceneGraph applications simply by adding the name of a Screen Graph scene defined in an XML component file, such as `posterScene`, as follows:

```
scene = screen.CreateScene("my_scene")
```

For example:

```
scene = screen.CreateScene("posterScene")
```

Similarly, you can control the flow of scenes through your application by creating and showing scenes as needed:

```
screen = CreateObject("roSGScreen")
m.port = CreateObject("roMessagePort")
screen.setMessagePort(m.port)
scene = screen.CreateScene("another_scene")
screen.show()
```

Set Up the components Directory

The `components` directory contains all the XML component and associated BrightScript code files needed for your SceneGraph scene. The XML files must have the extension `.xml`, and as usual, BrightScript code files must have the extension `.brs`.

Each XML component file contains a single **<component>** element that contains a specific SceneGraph node/element tree defining that component.

For example:

Example Scene Graph XML Component File

```
<?xml version="1.0" encoding="utf-8" ?>

<component name="rectangleScene" extends="Scene" >

<script type="text/brightscript" >
<![CDATA[


sub init()
  m.top.setFocus(true)
end sub
]]>
</script>

<children>
<Rectangle
  id="bottomRectangle"
  color="0x0000FFFF"
  width="1280"
  height="60"
  translation="[0,620]"
/>
</children>

</component>
```

In the above example, the SceneGraph component is a definition of a **Scene** node class named `rectangleScene`. The component definition consists of a **<script>** element, which defines some BrightScript code to be used to initialize `rectangleScene`, and a **Rectangle** node definition, that defines the location, size, and color of a rectangle to be shown on the display screen, with a node ID of `bottomRectangle`.

Complete information on creating XML component files and associated BrightScript code for your SceneGraph application can be found in:

- **SceneGraph Core Concepts**
- **SceneGraph Samples**
- **SceneGraph API Reference**

Set Up the images Directory

Any graphic image files to be included in the application package itself should be copied into the `images` directory. As a minimum, the `images` dire ctory *must* contain the application selection and splash screen graphic files described in **Create or Modify a manifest File** above. Other graphic files used in the application that will not be downloaded from your server should also be copied into the `images` directory.