

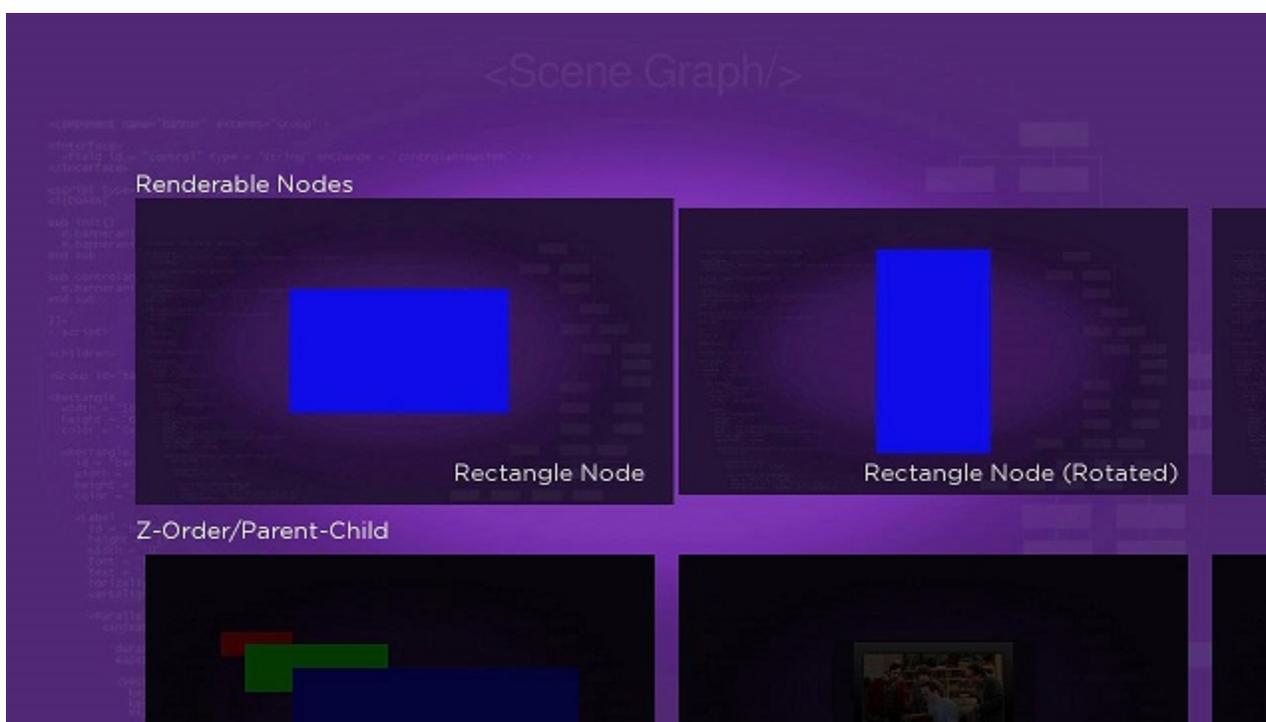
RowList Markup

Example Application: [RowListExample.zip](#)

Node Class Reference: [RowList](#)

The **RowList** node class incorporates characteristics of both lists and grids, and allows the same type of custom appearance and behavior of the list items (and the list rows) as in the **MarkupList** and **MarkupGrid** nodes.

In `RowListExample.zip`, we create a list of independently scrollable rows of grid items (similar to **PosterGrid** node items). As the user presses the **Left** and **Right** remote control keys, the row of grid items in the top row position scrolls to the left or right, and the focus is always set on the left grid item image, which enlarges slightly to indicate focus. Rows of grid items not in the top row position are dimmed. If the user presses the **Up** or **Down** remote control keys, the list of grid item rows scrolls up and down, and as each list row enters the top row position, the row grid items become fully visible, while the grid items in the other rows are always dimmed.



If you're wondering how that is done, you might want to review [MarkupList Markup](#) and [MarkupGrid Markup](#). That's because the same methods for creating custom lists and grids are blended together for the **RowList** node class, which combines characteristics of both lists *and* grids.

You can see this combination of list and grid in the `<children>` element of the `rowlistscene.xml` component file, where the example **RowList** node is defined:

Example RowList node defined

```

<children >
  <RowList
    id = "exampleRowList"
    translation = "[ 130, 160 ]"
    itemComponentName = "RowListItem"
    numRows = "4"
    itemSize = "[ 1608, 308 ]"
    rowItemSize = "[ [536, 308] ]"
    itemSpacing = "[ 0, 40 ]"
    showRowLabel = "[ true ]"
    drawFocusFeedback = "false"
    vertFocusAnimationStyle = "fixedFocusWrap"
    rowFocusAnimationStyle = "fixedFocusWrap" />
</children>

```

You can see some of the same fields used for both lists and grids used in this definition, and you can compare the definition with the equivalent definitions in [LabelList Markup](#), [PosterGrid Markup](#), [MarkupList Markup](#), and [MarkupGrid Markup](#).

Likewise, you might want to compare the item component definition in `rowlistitem.xml` with the equivalent `markuplistitem.xml` and `markupgriditem.xml` item component definitions:

RowList Item Component Example[Expand](#)

```

<component name = "RowListItem" extends = "Group" >
  <interface >
    <field id = "itemContent" type = "node" onChange = "showcontent" />
    <field id = "focusPercent" type = "float" onChange = "showfocus" />
    <field id = "rowFocusPercent" type = "float" onChange = "showrowfocus" />
  </interface>

  <script type = "text/brightscript" >

    <![CDATA[

    sub init()
      m.itemposter = m.top.findNode("itemPoster")
      m.itemmask = m.top.findNode("itemMask")
      m.itemlabel = m.top.findNode("itemLabel")
    end sub

    sub showcontent()

```

[source](#)

```
        itemcontent = m.top.itemContent
        m.itemposter.uri = itemcontent.HDPosterUrl
        m.itemlabel.text = itemcontent.title
    end sub

    sub showfocus()
        scale = 1 + (m.top.focusPercent * 0.08)
        m.itemposter.scale = [scale, scale]
    end sub

    sub showrowfocus()
        m.itemmask.opacity = 0.75 - (m.top.rowFocusPercent * 0.75)
        m.itemlabel.opacity = m.top.rowFocusPercent
    end sub

    ]]>
</script>

<children >

    <Poster
        id = "itemPoster"
        translation = "[ 10, 10 ]"
        width = "512"
        height = "288"
        scaleRotateCenter = "[ 256.0, 144.0 ]" >

        <Rectangle
            id = "itemMask"
            width = "512"
            height = "288"
            scaleRotateCenter = "[ 256.0, 144.0 ]"
            color = "0x101010FF"
            opacity = "0.75" />

    </Poster>

    <Label
        id = "itemLabel"
        translation = "[ 20, 264 ]"
        width = "492"
        horizAlign = "right"
        opacity = "0.0" />
```

```
</children>  
  
</component>
```

Compared to the other two examples with custom item component definitions, we've added a new **<interface>** field, `rowFocusPercent`, that triggers the `showrowfocus()` callback function when the field value changes. We are using the same basic "animations" as before based on that field and the `focusPercent` field value, but this time to only dim the unfocused rows, and only enlarge the focused item:

Dimming unfocused rows and enlarging focused item

```
sub showfocus()  
  scale = 1 + (m.top.focusPercent * 0.08)  
  m.itemposter.scale = [scale, scale]  
end sub  
sub showrowfocus()  
  m.itemmask.opacity = 0.75 - (m.top.rowFocusPercent * 0.75)  
  m.itemlabel.opacity = m.top.rowFocusPercent  
end sub
```