Introduction to Mobile App Development

MODULE 3: User Interface: Basic Concepts

Module 3

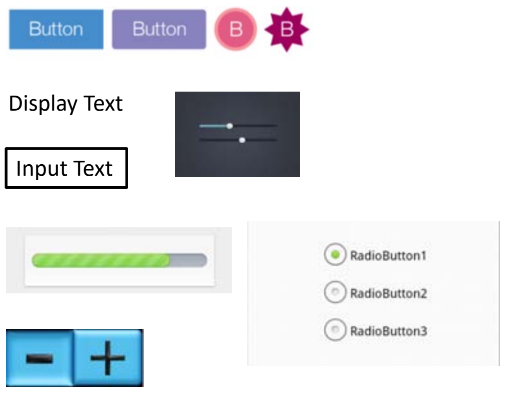
1. Introduction to User Interface Design
2. Corona Libraries
3. Buttons
4. Display Text
5. Input Text

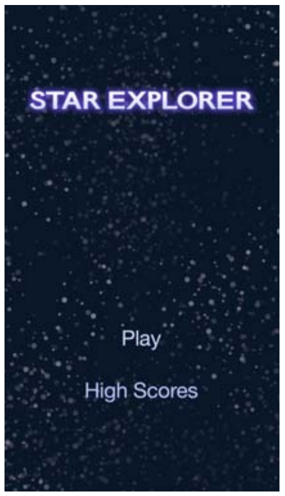
Introduction to User Interface Design

User Interface, Scenes and Widgets

Introduction

- A **scene** is an isolated view or "page" of the app and everything that the player sees is contained in that scene.





Corona Libraries

Corona Libraries

29 Libraries

- ads.*
- audio.*
- composer.*
- crypto.*
- display.*
- easing.*
- facebook.*
- gameNetwork.* (globals)
- graphics.*
- io.*
- json.*
- lfs.* (file system)
- licensing.*
- math.*
- media.*
- native.*
- network.*
- os.*
- package.*
- physics.*
- socket.*
- sqlite3.* (database)
- store.*
- string.*
- system.*
- table.* (array)
- timer.*
- transition.*
- widget.*

Buttons

7

Buttons

Library: Widget

Syntax: widget.newButton(options)

This function takes a single argument, options, which is a table that accepts the following basic parameters:
id (optional)
String. An optional identification string to assign for the button. Default is widget_button.
x, y (optional)
Numbers. Coordinates for the widget's x and y center point. These values will be overridden by left and top if those values are defined.
left, top (optional)
Numbers. The left and top position where the widget will be created. If specified, these values override the x and y parameters.
isEnabled (optional)
Boolean. If false, the button will not respond to touch events. Use button:setEnabled() to enable or disable touch events on a button after creation. Default is true (button is enabled).
onPress (optional)
Listener. An optional function to be called when the button is pressed. The callback function does not require testing for event.phase since it only honors "began".
onRelease (optional)
Listener. An optional function to be called when the user releases the button (assuming the touch is still over the button). The callback function does not require testing for event.phase since it only honors "ended".
onEvent (optional)
Listener. An optional function that should only be specified if onPress and onRelease are not set. This callback function allows you to test for the event.phase of "began", "moved", or "ended".

Reference: https://docs.coronalabs.com/api/library/widget/newButton.html

8

Buttons and Texts

Button Visual Options

- **Default**
- **2-Image Construction**
- **2-Frame Construction**
- **9-Slice Construction**

Reference: https://docs.coronalabs.com/api/library/widget/newButton.html

Buttons and Texts

Button Example 1: Default

```
local widget = require( "widget" )

-- Function to handle button events
local function handleButtonEvent( event )

   if ( "ended" == event.phase ) then
      print( "Button was pressed and released" )
   end
end

-- Create the widget
local button1 = widget.newButton(
   {
    x = display.contentCenterX,
    y = display.contentCenterY,
    id = "button1",
    label = "Default",
    onEvent = handleButtonEvent
   }
)
```

Reference: https://docs.coronalabs.com/api/library/widget/newButton.html

Buttons and Texts

Button Example 2: 2-Image

```
local widget = require( "widget" )

-- Function to handle button events
local function handleButtonEvent( event )

   if ( "ended" == event.phase ) then
      print( "Button was pressed and released" )
   end
end


 playBtn = widget.newButton{
      id = "playbutton",
      label = "Play",
      labelColor = { default={ 1, 1, 1 }, over={ 0, 0, 0, 0.5 } },
      emboss=true,
      width = 100,
      height = 100,
      fontSize = 30,
      defaultFile = "buttonDefault.png",
      overFile = "button_Over.png",
      onEvent = handleButtonEvent
   }
-- Center the button
playBtn.x = display.contentCenterX
playBtn.y = display.contentCenterY
```

Reference: https://docs.coronalabs.com/api/library/widget/newButton.html

Display Text

Library: Display

Syntax: display.newText(options)

https://docs.coronalabs.com/api/library/display/newText.html

This function takes a single argument, options, which is a table that accepts the following parameters:
parent (optional)
GroupObject. Display group in which to insert the text object.
text (required)
String. The text to display. Similarly, to change the displayed text for a text object after it has been created, set the object.text property.
x, y (optional)
Numbers. Coordinates for the object's x and y center point.
width, height (optional)
Numbers. If supplied, text will be wrapped at width and cropped at height. Set height to 0 and the text box height will adjust to the amount of text, but never exceed the maximum texture height for the device.
font (required)
String, Userdata, or Constant. This can be one of the following:

- The font family name (typeface name). You may obtain an array of available font names via native.getFontNames().
- Name of the font file in the Corona project's main resource directory (alongside main.lua).
- A font object returned by native.newFont().
- A font constant such as native.systemFont or native.systemFontBold.

fontSize (optional)
Number. The size of the text in Corona content points. The system's default font size will be used if this parameter is omitted or if it's set to nil or 0.
Important: To change the font size of a text object after it has been created, set the object.size property, not object.fontSize.
align (optional)
String. This specifies the alignment of the text when the width is known, meaning it either contains a newline or the width parameter is supplied. Default value is "left". Valid values are "left", "center", or "right".

Display Text

Example: 1

- **Single-Line Text**

```
local myText = display.newText( "Hello World!", display.contentCenterX, display.contentCenterY-100, native.systemFont, 30 )
myText:setFillColor( 1, 1, 0 )
```

- **Updating Text Post-Creation**

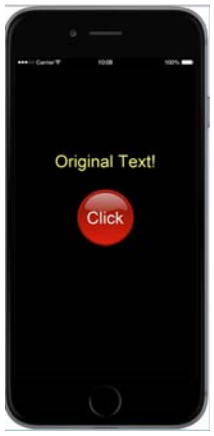
```
local myText = display.newText( "hello", 50, 50, native.systemFont, 12 )
myText:setFillColor( 1, 0, 0.5 )

-- Change the displayed text
myText.text = “New Text"

-- Increase the font size
myText.size = 16
```

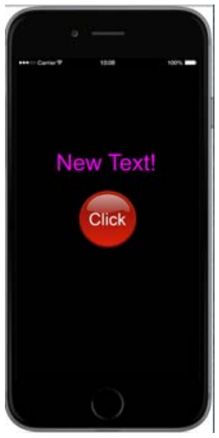
Reference: https://docs.coronalabs.com/api/library/display/newText.html

Buttons and Text

Example: 2

```
local widget = require( "widget" )
local counter = 0
local myText = display.newText( "Original Text!", display.contentCenterX, display.contentCenterY-100, native.systemFont, 30 )

myText:setFillColor( 1, 1, 0.5)


-- Function to handle button events
local function handleButtonEvent( event )

   if ( "ended" == event.phase ) then
     counter = counter + 1
      print( "Button was pressed and released" )
      print (counter)
      if ( counter % 2 == 0) then
      myText.text = "Original Text!"
      myText.size = 30
      myText:setFillColor( 1, 1, 0.5 )
      else
      myText.text = "New Text!"
      myText.size = 40
      myText:setFillColor( 1, 0, 1 )
      end
   end
end

 clickbutton = widget.newButton{
      id = "clickbutton",
      label = "Click",
      labelColor = { default={ 1, 1, 1 }, over={ 0, 0, 0, 0.5 } },
      emboss=true,
      width = 100,
      height = 100,
      fontSize = 30,
      defaultFile = "buttonDefault.png",
      overFile = "button_Over.png",
      onEvent = handleButtonEvent
   }
-- Center the button
clickbutton.x = display.contentCenterX
clickbutton.y = display.contentCenterY
```





Input Text

Input Text

Library: Native

https://docs.coronalabs.com/api/library/native/newTextField.html

Syntax: native.newTextField(centerX, centerY, width, height)

centerX, centerY (required)
Numbers. The x and y coordinates that correspond to the center of the text field.
width, height (required)
Numbers. Width and height (size) of the text field.

This creates a single-line text input field, typically used for gathering smaller bits of text (names, passwords, key codes, etc.).

https://docs.coronalabs.com/api/library/native/newTextBox.html

Syntax: native.newTextBox(centerX, centerY, width, height)

centerX, centerY (required)
Numbers. The x and y coordinates that correspond to the center of the text box.
width, height (required)
Numbers. Width and height (size) of the text box.

This creates a multi-line text input box of an arbitrary width and height, typically used for gathering longer pieces of text content.

17

Input Text

Example: Text Field

```
local defaultField

local function textListener( event )

   if ( event.phase == "began" ) then
      -- User begins editing "defaultField"

   elseif ( event.phase == "ended" or event.phase == "submitted" ) then
      -- Output resulting text from "defaultField"
      print( event.target.text )

   elseif ( event.phase == "editing" ) then
      print( event.newCharacters )
      print( event.oldText )
      print( event.startPosition )
      print( event.text )
   end
end

-- Create text field
defaultField = native.newTextField( 150, 150, 180, 30 )
defaultField:addEventListener( "userInput", textListener )
```

18

Input Text

Example: Text Box

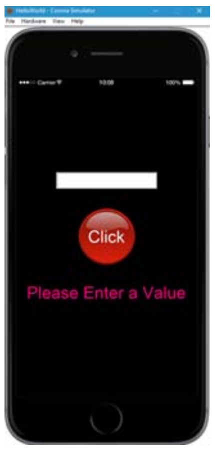
```
local defaultBox

local function textListener( event )

   if ( event.phase == "began" ) then
      -- User begins editing "defaultBox"

   elseif ( event.phase == "ended" or event.phase == "submitted" ) then
      -- Output resulting text from "defaultBox"
      print( event.target.text )

   elseif ( event.phase == "editing" ) then
      print( event.newCharacters )
      print( event.oldText )
      print( event.startPosition )
      print( event.text )
   end
end

-- Create text box
defaultBox = native.newTextBox(150, 200, 180, 140)
defaultBox.text = "This is line 1.\nAnd this is line2"
defaultBox.isEditable = true
defaultBox:addEventListener( "userInput", textListener )
```

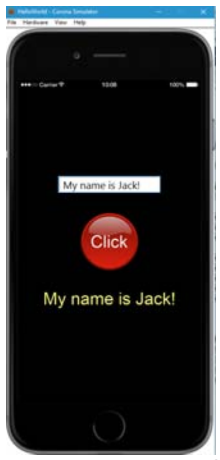
19

Buttons and Texts

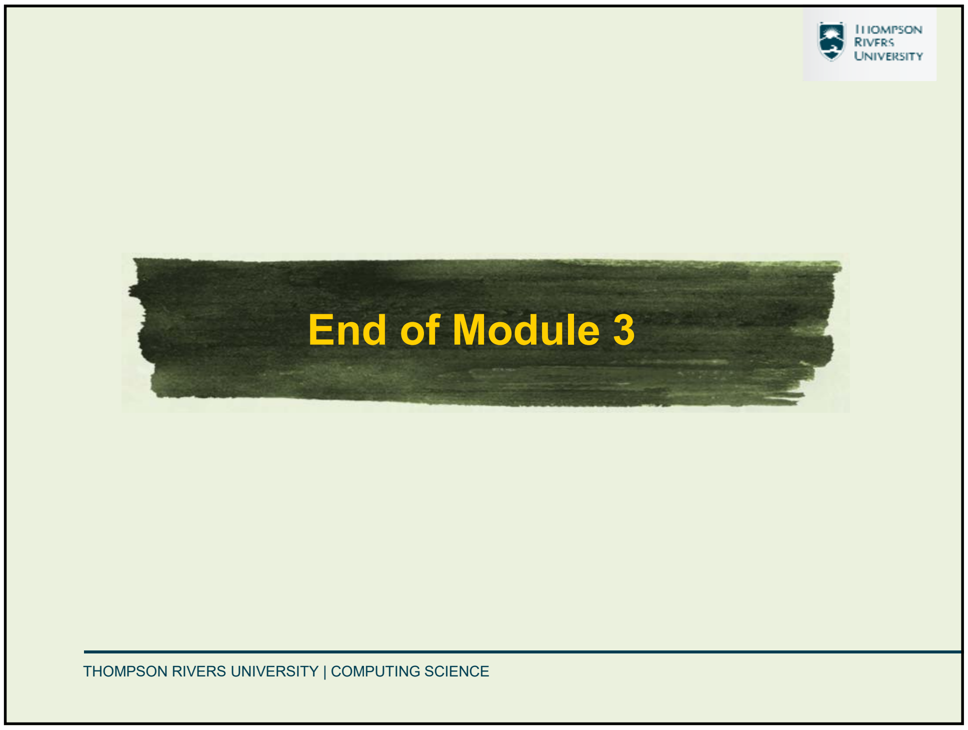
Example: A simple UI

```
local widget = require( "widget" )
local counter = 0
local myText = display.newText( "Original Text!", display.contentCenterX, display.contentCenterY+100, native.systemFont, 30 )
local inputText = native.newTextField( display.contentCenterX, display.contentCenterY-100, 180, 30 )

myText:setFillColor( 1, 1, 0.5)

-- Function to handle button events
local function handleButtonEvent( event )

   if ( "ended" == event.phase ) then
     if (inputText.text == "") then
        myText.text = "Please Enter a Value"
        myText:setFillColor( 1, 0, 0.5)
     else
        myText.text = inputText.text
        myText:setFillColor( 1, 1, 0.5)
     end
   end
end

 clickbutton = widget.newButton{
      id = "clickbutton",
      label = "Click",
      labelColor = { default={ 1, 1, 1 }, over={ 0, 0, 0, 0.5 } },
      emboss=true,
      width = 100,
      height = 100,
      fontSize = 30,
      defaultFile = "buttonDefault.png",
      overFile = "button_Over.png",
      onEvent = handleButtonEvent
   }
-- Center the button
clickbutton.x = display.contentCenterX
clickbutton.y = display.contentCenterY
```





20



End of Module 3

THOMPSON RIVERS UNIVERSITY | COMPUTING SCIENCE