TINKER ACADEMY

Mobile App Development

Handout 4: Touch Event Listeners User Interface

Note your Student ID. You will need to use it throughout the Course.

Setup Instructions In Classroom

Connect to the Local Class Network

1. Select WiFi “TINKER ACADEMY”
2. This network has only LOCAL access and does NOT connect to the internet

Update the Course

1. Ensure you are connected to “TINKER ACADEMY”
2. Restart the VM. Login into the VM.
3. Open Firefox in the VM
4. Your Instructor would tell you what to type in the browser. (Typically it is 192.168.1.5)
5. You should see a page with a list of entries.
6. Click on CourseUpdate<Date>.zip. This will download CourseUpdate<Date>.zip onto your VM
7. Open Nautilus. Click on Downloads. You should see the file CourseUpdate<Date>.zip
8. Right Click on CourseUpdate<Date>.zip. Select Extract Here.
9. Open the extracted folder
10. Double click Course Update. Select “Run” in the window.

Update the Course (Alternate Approach In Class Using USB)

1. Borrow a USB drive from the Instructor
2. If you are on VirtualBox
    a. Click on Devices in the Top level Menu
    b. Select Drag ‘n’ Drop
    c. Select Bidirectional
3. If you are on VirtualBox (Another Way)
    a. Shutdown Virtual Machine
    b. Click on VM in the VirtualBox Manager
    c. Click on the Settings
    d. Click General
    e. Click Advanced Tab

    f. Select “Bidirectional” under Drag ‘n’ Drop
    g. Click OK
    h. Start Virtual Machine
4. If you are on VMWare
    a. Open the virtual machine settings editor (VM > Settings),
    b. Click the Options tab
    c. Select Guest isolation.
    d. Deselect Disable drag and drop to and from this virtual machine
5. Open Nautilus, Click on Desktop
6. Drag the file CourseUpdate<Date>.zip from Windows or Mac onto Desktop in your Virtual Machine
7. Right Click on CourseUpdate<Date>.zip. Select Extract Here.
8. Open the extracted folder
9. Double click Course Update. Select “Run” in the window.
10. Eject the USB Drive and hand it back to the Tinker Academy instructor

Setup Instructions At Home

Connect to your Home WiFi Network

Updating the Course (Using Wifi)

1. Make sure you are on the Home WiFi Network.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Update”. Choose “Run”.
   If you see a window popup with the message “update course failed”.
   Hop onto Skype, and request help in the class chat group.
   And send an email to classes@tinkeracademy.com with your name and student ID.
4. Follow the instructions in this handout (last 2 pages) on the quiz and homework steps.

Submitting Quiz and Homework

1. Make sure you are on the Home WiFi Network.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Submit”. Choose “Run”.
   If you see a window popup with the message “submit course failed”.
   Hop onto Skype, and request help in the class chat group.
   And send an email to classes@tinkeracademy.com with your name and student ID.

Virtual Machine Installation

Installing the Virtual Machine (VM)

1. Borrow the USB drive from your Tinker Academy instructor

2. Create the folder “tinkeracademy” (without the quotes) under Documents using Finder or Windows Explorer. Type it in *exactly* as indicated.
3. Copy the folder “installers” from the USB drive to under “tinkeracademy” using Finder or Windows Explorer
4. Eject the USB Drive and hand it back to the Tinker Academy instructor
5. Locate the VirtualBox installer under “tinkeracademy” using Finder or Windows Explorer

If your Laptop is	Double click on
Windows 7	VirtualBox-4.3.12-93733-Win.exe
Windows 8	VirtualBox-4.3.14-95030-Win.exe
Mac	VirtualBox-4.2.26-95022-OSX.dmg

6. Install the VirtualBox application
7. Congratulations, You completed a major milestone. Give yourself a pat on the back :)

Importing the Virtual Machine (VM)

1. Locate the Virtual Machine “tinkeracademy.ova” under “tinkeracademy”
2. Double click on “tinkeracademy.ova”. You should get the import screen in VirtualBox with an “Import” Button. Click on the “Import” button to Import the Virtual Machine.

Starting the Virtual Machine (VM)

1. Once the Import is complete and successful, you should see the VM “TinkerAcademy” in the side panel in VirtualBox.
2. If it says “Powered Off” click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
3. If it says “Running” click on the Show Button (Green Arrow) in the VirtualBox Toolbar. This should display the VM window.
4. Once the VM starts up you will be presented with a login screen. Type in “password” without the quotes. Type it in exactly as indicated and hit “Enter”.
5. Once the login is completed you should see a Desktop with a few icons. The Screen might go fuzzy for a few seconds before displaying the Desktop. *That is ok.*
6. Congratulations. You are now running Linux within your laptop.
7. Double click on the “Firefox” icon in the Sidebar. This should launch Firefox. Verify you have network access. Close “Firefox”

Launching the Virtual Machine in Full Screen

1. Use the VirtualBox menu View->Switch to Fullscreen to switch the VM to fullscreen mode
2. Use the same VirtualBox menu View->Switch to Fullscreen to switch the VM back out of fullscreen mode

Shutting Down the Virtual Machine

1. Click on the red close window button (to the top left on a Mac, top right in Windows).
2. You will prompted with a confirmation message asking if you want to “Power Off” the machine. Click the button to confirm power off.
3. In a few minutes the VM will shut down and you should see the VirtualBox side panel with the “Tinker academy” VM indicating “Powered Off”.

Restarting the Virtual Machine

1. Start VirtualBox
2. Click on the VM “TinkerAcademy” in the VirtualBox side panel.
3. Click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
4. Once the VM startup you will be presented with a login screen.

Right Click in VM on Mac

1. Open System Preferences, Trackpad
2. Enable “Secondary Click”, Toggle the small arrow to the right and select “Click with two fingers”.

Installing Corona SDK and Sublime Text

1. Ensure you are connected to “TINKER ACADEMY”
2. Open **Chrome or Safari in Windows or Mac** (NOT in the VM)
3. Your Instructor would tell you what to type in the browser. (Typically it is 192.168.1.5)
4. You should see a page with a list of entries.
5. Click on TA-GME-1-Installers.zip. This will download TA-GME-1-Installers.zip onto your VM
6. Extract TA-GME-1-Installers.zip.
7. Open the extracted folder
8. Open corona.
9. Select the installer ending in .dmg for Mac.
10. Select the installer ending in .msi for Windows.
11. Double click to begin the installation process for Corona.
12. Open sublimetext.
13. Select the installer ending in .dmg for Mac.
14. Select the installer ending in Setup.exe for Windows. If you are on a 64 bit Windows select the installer ending in x64 Setup.exe. If you are not sure if you are on 64 bit Windows select Setup.exe.
15. Double click to begin the installation process for Sublime Text 2.

Register Corona SDK

1. Ensure you are connected to the Cupertino WiFi
2. If your Skype ID is tinkeracademy001 use student001@tinkeracademy.com as your email address. Use tinker2014 as your password

Enable Drag ‘n’ Drop

1. If you are on VirtualBox
    a. Click on Devices in the Top level Menu
    b. Select Drag ‘n’ Drop
    c. Select Bidirectional
2. If you are on VirtualBox (Another Way)
    a. Shutdown Virtual Machine
    b. Click on VM in the VirtualBox Manager
    c. Click on the Settings
    d. Click General

e. Click Advanced Tab
f. Select “Bidirectional” under Drag ‘n’ Drop
g. Click OK
h. Start Virtual Machine

3. If you are on VMWare
    a. Open the virtual machine settings editor (VM > Settings),
    b. Click the Options tab
    c. Select Guest isolation.
    d. Deselect Disable drag and drop to and from this virtual machine

Corona Simulator

1. Open Corona Simulator
2. Run the demo programs

About Corona

What is Corona?

Corona is an extremely powerful, easy to program Mobile Application Framework.
You can use Corona to develop powerful applications that can then be made to run on differe
mobile devices, iOS (iPhone, iPad), or Android (phone, tablets)

You can also use Corona to develop 2D games.

Corona uses Lua as the programming language used to build Mobile Apps and Games.

What is a Game Engine?

A game engine is a software framework designed for the creation and development of video games.

A 2D Game engine provides support for creating Sprite based games. Sprites are 2D images

The Game engine provides a rendering engine to display the graphics, physics engine to simulate realistic physics, sound, animation, networking.

The Game Engine provides support for all these features through the Software Development Kit as APIs or Application Program Interfaces.

APIs are nothing but Corona modules that have tables, functions and other Lua code that we can invoke from our program.

About the Corona Simulator

What is the Corona Simulator?

Mobile Applications developed meant to be run on mobile devices. Mobile Application Frameworks such a Corona work very hard to simply development by providing a simulator o the actual device. This Corona applications and games to be run on the computer while simulating the look and feel of being run on mobile devices.

The Corona Simulator allows simulating different devices.

1. Open Corona Simulator
2. Click on Window in the menu bar.
3. Select View As, <Device> to switch to a different device
4. Use the Window, Zoom In, Zoom Out, Minimize and Center controls to adjust the width and height of the simulator on your laptop.

We will be using the Simulator View As Device set to iPhone for the duration of this course (unless specified otherwise)

Getting Started With Corona

Extract StarterPack4.zip
StarterPack4 is an empty Corona Project. We will be using StarterPack4 to create a physics scene.

Layout of a Corona Application Project

Corona Applications exist as folders.

StarterPack4 represents an empty Corona Project.

A Corona project is typically made up of the following

1. A required main.lua lua file
2. An optional config.lua config file
3. An optional build.settings file
4. Zero or more additional .lua files
5. Zero or more assets, typically image assets as PNG files

The Main Lua File

Every Corona project is required to have a file named main.lua.

Apart from the name, there is nothing special about the main.lua file. The file is a Lua chunk. The structure of the file is similar to that of the Lua files we created in Handout 2.

The Config Lua File

The config lua file is a Lua file that contains specific code for the project configuration.

The Corona application will need to run on multiple mobile devices.

The config file provides support to specify

- Common width and height. You can then use the width and height in your program. Corona will automagically scale the application to fit the device screen size

The Build Settings File

The build settings file is used to control how the application project will get built to different mobile devices.

What is building?

Corona Mobile Applications are written in Lua. However Lua is not a language that iOS devic or Android devices understand.

Building is the step where the Corona application gets converted using various steps into an application that either the iOS devices or the Android can understand.

The build settings file provides support to specify

- Device orientation “portrait” or “landscape”

The Corona Coordinate System

The Corona application is displayed in a physical Screen.
However Corona operates on a logical screen called the Content Area.

Think of the content area as the Stage in SCRATCH.

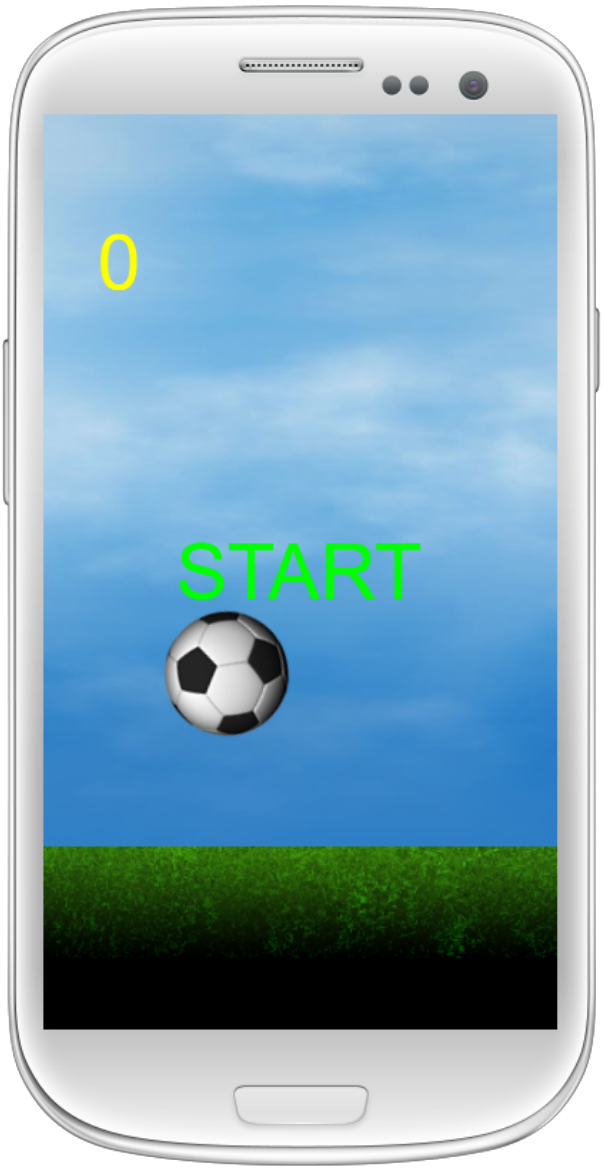
The width and height of the content area can be specified in the config.lua.

Corona assumes (by default) that the (0,0) coordinate is at the top left of the content area.
Corona will automatically scale the content area to fit the actual physical screen.

Getting Ready to Program

1. Launch Corona Simulator
2. Select Device “Galaxy S3”. Zoom In or Zoom out if necessary.
3. Open **simulplay** in Corona Simulator
4. You should see an empty screen. That is ok. We are going to add code shortly.

SimulPlay Part 1: The Scene



Add the Sky

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
local sky = display.newImage( "bkg_clouds.png", 160, 195 )
```

4. Save the file
5. The Corona Simulator should now display the sky as the background
6. Thats it! in a few keystrokes you created a sky background
7. Save File, Relaunch in Corona Simulator (Command-R on Mac, Ctrl-R Windows) if you missed all the fun

The main.lua is the first chunk of code that gets executed. The code reads a new image file, displays it and returns the image display object as the initial value of the local variable “sky”.

Think of display as a Lua table.

What is a Lua table?

Corona Mobile Applications are written in Lua.

A Lua table is an associative array.

Arrays are indexed by an indexes.

Associative arrays can be indexed by any valid Lua value, except nil.

(nil in Lua is the equivalent of null in Java)

The display table has a function newImage.

The function newImage that takes as input the image name, the x and y coordinates of the image on the Screen.

The function then reads the image data from the image file and displays the image in the content area at the location specified by the x and y coordinate.

Remember that the (0,0) coordinate is at the top left.

x coordinate increase towards the right.

y coordinate increase towards the bottom.

Add the Ground

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
local ground = display.newImage( "ground.png", 160, 445 )
```

3. Save File, Relaunch in Corona Simulator (Command-R on Mac, Ctrl-R Windows)

The code does something similar to the previous code. The ground.png PNG file is read, the image display object is constructed and placed at location (160, 445). The image display object is assigned to the local variable “ground”.

Start Your Physics Engines

Corona comes with a simple to use, extremely powerful physics engine. The physics engine is provided as a Corona Module.

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
local physics = require( "physics" )
physics.start()
```

The code loads the “physics” module using the require function.

The require function is provided by Corona to load modules.

physics.start() starts the physics simulation.

This makes it possible for this application to simulate physics effects such as Gravity.

Add Objects To The Physics Engine

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
physics.addBody( ground, "static", { friction=0.5, bounce=0.3 } )
```

The code requests the physics engine to treat the ground image as a static object. A static object would not move due to gravity. This is ok! since the ground should not move due to gravity.

The code then indicates that the ground object has a friction of 0.5 units (out of 1) and a bounce of 0.3.

What is friction?

Friction is the force that is created when 2 surfaces try to move.

In Corona a value of 0.5 indicates that the ground exerts a fair amount of friction on any objec that tries to move on its surface.

In Corona, a value of 0 means no friction and 1.0 means fairly strong friction. The default valu is 0.3.

What is bounce?

Bounce indicates how “bouncy” a surface is.

In Corona the value of 0.3 indicates slightly bouncy surface.

Values greater than 0.3 are fairly "bouncy," and an object with a bounce value of 1.0 will rebound forever.

Bounce values higher than 1.0 are valid, and they will actually gain velocity with each collisior The default value is 0.2, which is slightly bouncy.

Add More Objects To The Physics Engine

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
local ball = display.newImage( "soccer_ball.png", 180, -50 )
ball.rotation = 5
physics.addBody( ball, { density=3.0, friction=0.5, bounce=0.8 } )
```

The code does something similar to the rest of the code above. The soccer_ball.png image file is read and displayed at the location 180, -50. -50 indicates that the ball starts off being not visible in the screen.

The ball also has a rotation speed set to 5. This will mean that the ball will rotate slightly if it hits a surface such as the ground.

The next line of code adds the ball object to the physics engine. The ball object is “dynamic” (the default if static is not specified). A “dynamic” object would move due to gravity. The object has a friction and bounce and a density.

What is density?

Density indicates how heavy something is.

Air is light and has low density.

Gold is heavy and has high density.

In Corona, density is based on a standard value of 1.0 for water.

Lighter materials (such as wood) have a density below 1.0, and heavier materials (such as stone) have a density greater than 1.0. Default value is 1.0.

3. Save File, Relaunch in Corona Simulator (Command-R on Mac, Ctrl-R Windows)

Keeping Score

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
widget = require("widget")

scoreButton = widget.newButton
{
```

```
        id = "score",
        isEnabled = false,
        fontSize = 48,
        font = "Arial",
        x = 48,
        y = 48,
        textOnly = true,
        labelColor = { default={ 1, 1, 0 } }
}
```

3. Save File, Relaunch in Corona Simulator (Command-R on Mac, Ctrl-R Windows)

The code above adds a button to display the score.

Display the Score

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
function displayScore(score)
        scoreButton:setLabel(score)
end

displayScore(0)
```

3. Save File, Relaunch in Corona Simulator (Command-R on Mac, Ctrl-R Windows)

The code defines a new lua function displayScore. displayScore is a lua function that takes one input, score, and sets the label in the scoreButton.

The code then invokes displayScore with the initial score of 0.

Add the START Button

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
startButton = widget.newButton
{
        id = "start",
        label = "START",
```

```
        fontSize = 48,
        font = "Arial",
        x = 160,
        y = 240,
        labelColor = { default={ 0, 1, 0 }, over={0, 1, 0} }
}
```

3. Save File, Relaunch in Corona Simulator (Command-R on Mac, Ctrl-R Windows)

The code above adds the START button.

Add code to START the Game

1. Open main.lua in **simulplay** in Sublime Text
2. Type in the code below in main.lua

```
require "game"

function startButtonTappedListener(event)
        startButton:removeSelf()
         ball:removeSelf()
        startGame()
end

startButton:addEventListener("tap", startButtonTappedListener)
```

3. Save File, Relaunch in Corona Simulator (Command-R on Mac, Ctrl-R Windows)

require is a built-in in Lua. Here it takes one input “game”, which is the name of the Lua file “game.lua” that is required to be loaded.

What is a built-in function?

A built-in function in a programming language such as a Lua is a function provided as part of the language.

What is loading a Lua file?

When the require function is invoked with the file name, the Lua interpreter reads the Lua file and processes it. This process is called loading.

The code also defines an event listener. Event Listeners “listen” to events.

What is an event?

Corona listens to interesting occurrences in the program.

An event is the occurrence of something interesting in the program. Events are created by objects.

Many things can happen when the user interacts with the program.

User taps the screen. The tap touches a Button. The Button creates a “tap” Event.

What is an event listener?

Event listeners are functions. The function takes 1 input

1. The event

They are registered with the Lua object that creates the event.

When the Lua object creates the event, Lua ensures that all event listeners that have registered with the Lua object get invoked.

The code defines an event listener startButtonTappedListener to “listen” to the touch event on the startButton.
As soon as the user “taps” on the start button, the startButtonTappedListener is invoked.

The startButtonTappedListener function does the following

1. invokes startButton:removeSelf()
   The removeSelf function removes the startButton from the display
2. invokes the ball:removeSelf()
   The removeSelf() function removes the ball from the display
3. invokes the startGame function in the game.lua

Thats it! On saving the main.lua file and reloading the project in the simulator, you should see the ball fall from the sky due to gravity, hit the ground, rotate slightly and and bounce a few times before coming to a complete stop.

SimulPlay Part 2: The Game Logic

What is the game logic?

The objective of the game is to kick around a soccer ball without letting it collide with the ground. Multiple Players launch the game simultaneously. Each player gets a turn at random.

The Game starts on tapping the START button.

When its the player's turn to play, the game will drop the soccer ball from a random locati

The player has to "kick" the ball to ensure that the soccer ball does not collide with the ground

If the soccer ball is "kicked" successfully, all players get a score increased by 100 points.

If any player "misses" and the ball collides with the ground, then all player's score gets reset t 0.

Initialize The Game

1. Open game.lua in **simulplay** in Sublime Text

2. Type in the code below within the startGame function in game.lua

```
function onGameUpdate(score, shouldDropBall)
end

function startGame()
        initializeGame(onGameUpdate)
end
```

The code contains hooks up the code to initialize the game.

When the START button is clicked, the startGame function is invoked.

The startGame function invokes the function initializeGame.

The function initializeGame requires a function as the input that will be invoked whenever the game is updated.

The function initializeGame ensures that the game is initialized correctly and onGameUpdate is invoked whenever the game is updated.

(initializeGame is part of the Game API. We will cover it a later section.)

onGameUpdate takes 2 inputs

1. score
2. shouldDropBall

score refers to the current game score.
shouldDropBall indicates if the ball should be dropped into the game

Add in the Game Play

What is game play?

Game play refers to the the plot of the game.

In our game

When its the player's turn to play, the game will drop the soccer ball from a random location.

1. Open game.lua in **simulplay** in Sublime Text

2. Type in the highlighted code below within the startGame function in game.lua

```
function onGameUpdate(score, shouldDropBall)
	displayScore(score)
	if shouldDropBall then
		dropBall()
	end
end

function onBallTapped(event)
	kickBall()
```

```
end
```

displayScore, dropBall and kickBall are part of the Game API.

We will cover it in the next section.

Review the Game API

The Game Logic uses the following Game APIs

Game API	Inputs	It provides...
initializeGame	callback debug	• Initializes the game timer • Listens to collision events • Registers the onGameUpdate function
dropBall		• creates a ball at a random location • adds physics to the ball • applies random “drop” physics force to the ball • listens to ball tap events
kickBall		• applies a random “kick” physics force to the ball

Review the Network API

The Game API uses the Network APIs to communicate with the server and receive game updates.

The network API’s communicate with server using the user id. The user id is a special generated id that is is guaranteed to be unique to within high levels of probability.

Network API	Inputs	It returns...
nextturn		• the user id of the next turn player • the latest group score

	user id	
addscore	user id	● adds to the group total score ● returns the latest group score
resetscore	user id	● resets the group score to 0 ● returns the latest group score

Put your Gaming Hats On!

Trial Run

3 Volunteers

1. Relaunch Launch Game in Corona Simulator (Command R on Mac or Control+R on Windows)
2. Don’t click the start button yet
3. Wait for the Ready Signal

The Real Deal

1. Relaunch Launch Game in Corona Simulator (Command R on Mac or Control+R on Windows)
2. Don’t click the start button yet
3. Wait for the Ready Signal

That was a LOT we covered!

You made it this far! Awesome!

In the next class, we will be moving to Graphics Images and Animation Audio and Video using the Corona Game Engine.

Quiz 4: Touch Event Listeners User Interface

Open the Quiz

Make sure you are on the Home WiFi.
Follow the instructions in “Updating the Course” in this Handout.
Open Quiz4.odt under “Courses” “TA-GME-1” “quiz” “quiz4”

Complete the Quiz

1. Attempt each question. Type in the answers in the “Answer:” box.
2. Save the file using File->Save or Ctrl-S

Submit the Quiz

Make sure you are on the Home WiFi.
Follow the instructions in “Submitting Quiz and Homework” in this Handout.