Corona

Coding a game with the Ansca Corona SDK

Presented By: David E. Rodriguez

Introduction to Corona

"A mobile development framework for creating high-performance, multimedia rich applications and games for the iPhone, iPad, and Android."

Introduction to Corona

- Cross Platform. Code once,
  deploy on iPhone, iPad, and Android!
- Programmed In Lua
- Free to try.
  Subscription based if you want to publish.

Tools

- Corona: **http://www.anscamobile.com/**
- Text Wrangler: **http://www.barebones.com/products/textwrangler**
- Physics Editor: **http://www.physicseditor.de/**
- Corona Remote: **http://www.coronaremote.com/**

Lua - Introduction

- Couple of Hints:
  - http://developer.anscamobile.com/content/introduction
  - print(hello) is your best friend
  - --[[Commented Code --]] or -- Comment
  - Don’t need a semicolon ;)

Lua - Types

Types and Values

- nil, boolean, number, string
- functions and tables
- local and global scope

Lua - Scope

```
x = 1                    -- global variable
local y = 10             -- local variable
```

- Global variables do not need declarations. They live as long as your application is running.
- Local variables are visible only in the block where they are declared.
- To delete a global variable, set it to nil.

Lua - Tables

- Tables are objects

```
t = {}                    -- create a table

k = "x"
t[k] = 3.14               -- new table entry, with key="x" and value=3.14
t[10] = "hi"              -- new table entry, with key=10 and value="hi"

print( t[k] )             --> 3.14
print( t["x"] )           --> 3.14
print( t.x )              --> 3.14

k=10
print( t[k] )             --> "hi"
```

Don’t confuse t.x with t[x]. The first is equivalent to t[“x”]

Lua - Table Fields

Table fields and properties.

```lua
t = { foo="hello" }                    -- create table with a single property "foo"
print( t.foo )                         --> "hello"

t.foo = "bye"                          -- assign a new value to property "foo"
print( t.foo )                         --> "bye"

t.bar = 10                             --> create a new property bar
print( t.bar )                         --> 10
```

Lua - Functions

- Most functions are stored in tables as properties.
  Example:
  Math is merely a `math.sin( 100 )` rty sin is the actual function.
- All of the Corona libraries such as the display library similarly group functions into tables.

Lua - Functions

```lua
function foo()
    -- Do one thing
end

function foo( arg1, arg2 )
    --[[ Do another thing --]]
end
```

Lua - Table Methods

```lua
Car = { location="home" }

function Car:drive( loc )
    self.location = loc
    --[[ Drive the car to a location --]]
end

Car:drive( "converge" )
```

- Methods defined with colon (:) operator
- Shortcut to this:

```lua
function Car.drive(self, loc)
    self.location = loc
    --[[ Drive the car to a location --]]
end
```

Lua

- Performs garbage collection
- No braces {} instead bracket code with **do** and **end**
- Arrays are 1-based.
  First element of a table is indexed by **t[1]** not **t[0]**.
- Multiple Assignment
  Example: **x,y = 0, 0** or **x, y = y, x** for a swap

Corona

- Corona provides simulator and debugger.
- Need one file to start a program: **main.lua**
- Android SDK and iOS SDK must be installed if you wish to compile cross platform.

Pick Your Poison

- Simple timed game
- Choose Gates or Jobs
- Tilt device to catch Windows or Apple Icons
- Rack up your score!

Pick Your Poison

Download Code

- programmerdave.tumblr.com
- Password for file is: davidrodriguez

Starting Code

```
---------- IMPORT MODULES ---------------
local physics = require("physics")
local audio = require("audio")

---------- MISCELLANEOUS  ---------------
math.randomseed( os.time(os.date("*t")) ) -- seed random number generator with the OS clock
display.setStatusBar( display.HiddenStatusBar ) -- Hide the status bar
```

Initialize Variables

```
---------- INITIALIZE VARIABLES  ---------------
local menu_img = nil -- On Screen Image Object
local background_img = nil -- On Screen Image Object
local player_img = nil -- On Screen Image Object

local scoreText = nil -- On Screen Text Object
local timeText = nil  -- On Screen Text Object
local timeLeft = 0
local gameScore = 0
local gameOver = false

local floor = nil -- An invisible rectangle that defines our floor
local player = "" -- This string is either "gates" or "jobs" depending on the character chosen

local gamePieceGroup = nil -- This DisplayGroup will keep all our game pieces in one place
local finalScoreText = nil -- On Screen Text Object
```

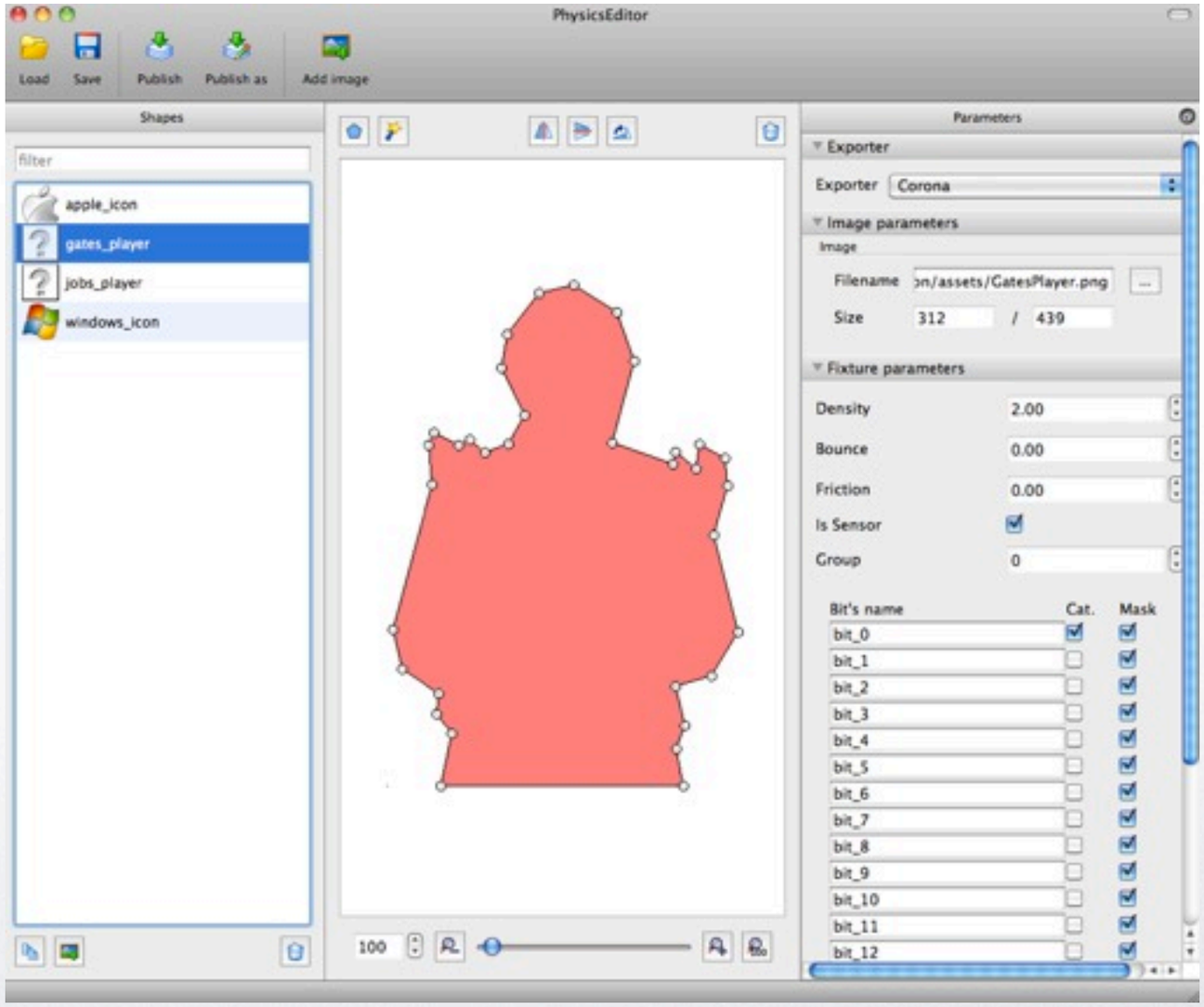
Initialize Variables

```
---------- LOAD SOUNDS  ---------------
local winSound = audio.loadSound("assets/win_sound.wav")
local loseSound = audio.loadSound("assets/lose_sound.wav")

---------- LOAD PHYSICS MODELS  ---------------
local physicsModels = (require "physicsModels").physicsData(1)
```

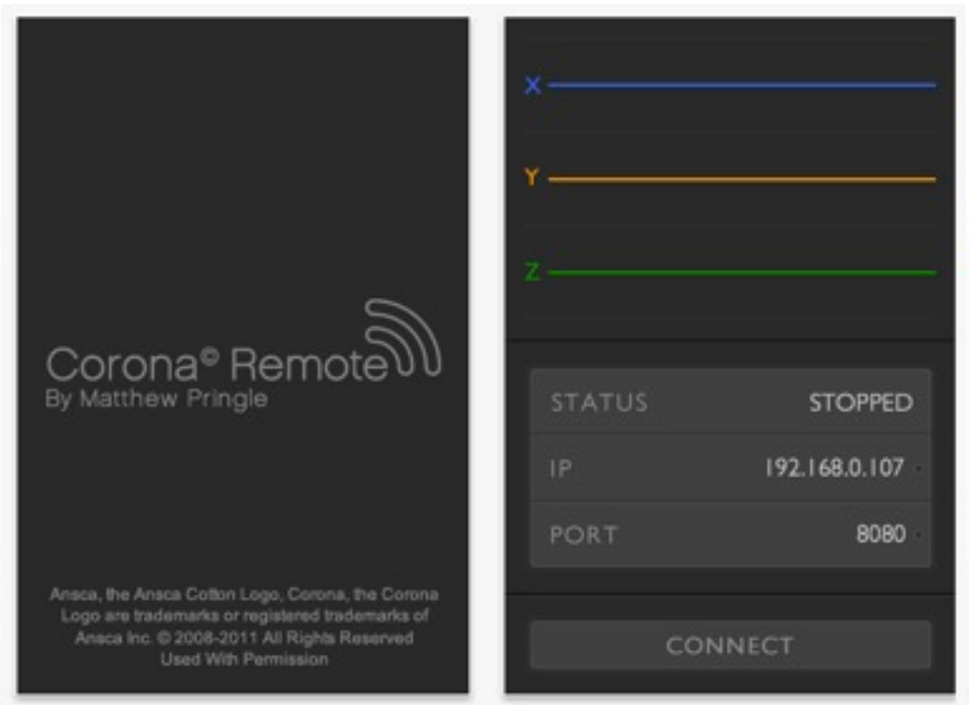
Physics Editor

- Simple to Use
- Import an Image, it creates a LUA code
- Call the function it created for you when you create the physics object



Corona Remote

- $9.99 in App Store
- Provides accelerometer data for app debugging
- Very easy to use
- Automatically reverts to hardware on device.




```
---------- CORONA REMOTE ---------------------
local remote = require("remote") -- Load The Remote
remote.startServer( "8080" ) -- Start The Remote On Port 8080
```

Functions - Creation

```lua
-- Create the score and remaining time display objects
function createText()

    scoreText = display.newText("Score: ", 10, 0, native.systemFont, 48)
    scoreText:setTextColor(255, 255, 255, 200)

    timeText = display.newText("Time Left: ", 10, 50, native.systemFont, 48)
    timeText:setTextColor(255, 255, 255, 200)
end
```

Functions - Creation

```
-- Creates an invisible rectangle for the floor
-- When an icon collides with this floor, it will be removed from the game
function createFloor ()

    floor = display.newRect( 0, 960, 640, 20 )
    floor:setFillColor(  255, 255, 255, 0)      -- make invisible by setting alpha to 0

    floor.collisionName = "floor" -- Name to use when detecting collisions

    -- add the physics model to the floor.
    -- This is a sensor that only detects collisions and doesn't simulate any physics
    physics.addBody( floor, "static", { isSensor = true } )
end
```

Functions - Creation

```
-- Create the player image and setup it's physics properties
function createPlayer( )

    -- load the player image
    if( player == "gates" ) then

        player_img = display.newImage( "assets/gates_player.png", true)

    else

        player_img = display.newImage( "assets/jobs_player.png", true)

    end

    -- Position the player
    player_img:setReferencePoint( display.BottomCenterReferencePoint )
    player_img.x, player_img.y = 320, 960

    player_img.collisionName = "player" -- Name to use when detecting collisions

    -- Add the physics model and make it kinematic so that we can control it
    if( player == "gates" ) then
        physics.addBody( player_img, "kinematic", physicsModels:get("gates_player"))
    else
        physics.addBody( player_img, "kinematic", physicsModels:get("jobs_player"))
    end

end
```

Functions - Creation

```lua
-- Spawns a Game Piece from thin air.
-- Randomly generated and positioned.
-- Called every second until the game is over.
function spawnGamePiece()

    if( gameOver == true) then
        return
    end


    local pick_icon = math.random()
    local icon_img = nil

    -- Choose either the windows icon or the apple icon
    if( pick_icon > 0.5 ) then -- Choose the windows icon

        icon_img = display.newImage( "assets/windows_icon.png", true) -- Load the image

        icon_img.collisionName = "windows" -- Name to use when detecting collisions

    else -- Choose the apple icon

        icon_img = display.newImage( "assets/apple_icon.png", true) -- Load the image

        icon_img.collisionName = "apple" -- Name to use when detecting collisions

    end
```

```lua
    -- Add the gamePiece to the gamePiece Display Group
    gamePieceGroup:insert(icon_img)

    -- Position icon into place randomly
    local x, y = math.random(120, 640 - 120), -60
    icon_img.x, icon_img.y = x, y

    -- Add Physics Model to the Icon
    if( pick_icon > 0.5 ) then -- Choose the windows icon

        physics.addBody( icon_img, physicsModels:get("windows_icon") ) -- add the physics model

    else -- Choose the apple icon

        physics.addBody( icon_img, physicsModels:get("apple_icon") ) -- add the physics model

    end

    timer.performWithDelay( 1000, spawnGamePiece ) -- Call this method again in a second
end
```

Functions - Update

```
-- Changes the text on screen with the current score and time remaining
function updateScoreAndTimeText()
    scoreText.text = "Score: " .. gameScore
    scoreText:setReferencePoint(display.CenterLeftReferencePoint)
    scoreText.x = 10

    timeText.text = "Time Left: " .. timeLeft
    timeText:setReferencePoint(display.CenterLeftReferencePoint)
    timeText.x, timeText.y = 10, 100
end
```

Functions - Update

```
-- This timer is called every second to keep track of the remaining time in the game
function updateGameTimer( event )

    timeLeft = timeLeft - 1 -- subtract a second from the remaining time

    if( timeLeft <= 0 ) then
        gameOver = true -- Done with game
    else
        updateScoreAndTimeText()

        timer.performWithDelay( 1000, updateGameTimer ) -- Call this method again in a second
    end
end
```

Functions - Update

```
-- Updates the game status and moves the player on every frame
function updateGame( event )

    if( gameOver == true ) then
        -- Show the score and wait 5 seconds before returning to the menu
        showFinalScore()

        -- Register to call gameTime's timer method every second
        timer.performWithDelay( 5000, endGame )
        return
    end

    -- Update the text on the screen
    updateScoreAndTimeText()

    -- Move character based on gravity
    player_img:translate( remote.xGravity*90, 0 )

    -- Keep the player inside the screen bounds
    if( player_img.x < -100 ) then
        player_img.x = -100
    end

    if( player_img.x > 640 + 100 ) then
        player_img.x = 640 + 100
    end

end
```

Functions - Collisions

```
-- This method is called when a collision occurs between any two objects
function onCollision( event )
    -- The collision just began
    if ( event.phase == "began" ) then

        -- Check if there was a collision between the floor and an icon
        if( event.object1.collisionName == "floor" or event.object2.collisionName == "floor" ) then

            -- If the icon collided with the floor, remove it from the display
            -- This will also remove it from the physics simulation
            if( event.object1.collisionName == "apple" or event.object1.collisionName == "windows" ) then

                print("Icon Disappeared")
                event.object1:removeSelf()

            elseif( event.object2.collisionName == "apple" or event.object2.collisionName == "windows" ) then

                print("Icon Disappeared")
                event.object2:removeSelf()

            end
```

Functions - Collisions

```
-- Check if there was a collision between the player and an icon
elseif( event.object1.collisionName == "player" or event.object2.collisionName == "player" ) then

    -- If the icon collided with the player, udpate the score, then remove it from display
    -- Also plays a sound and either increases or decreases the score based on the player type
    if( event.object1.collisionName == "apple" or event.object2.collisionName == "apple" ) then

        print("Player Captured Apple")

        -- Update Score
        if( player == "gates" ) then -- apples are bad for Bill Gates
            gameScore = gameScore - 1
            audio.play(loseSound)
        else
            gameScore = gameScore + 1
            audio.play(winSound)
        end

        -- Remove the icon from display
        if( event.object1.collisionName == "apple" ) then
            event.object1:removeSelf()
        else
            event.object2:removeSelf()
        end
```

```
        elseif( event.object1.collisionName == "windows" or event.object2.collisionName == "windows" ) then

            print("Player captured Windows")

            -- Update Score
            if( player == "jobs" ) then -- windows are bad for Steve Jobs
                gameScore = gameScore - 1
                audio.play(loseSound)
            else
                gameScore = gameScore + 1
                audio.play(winSound)
            end

            -- Remove the icon from display
            if( event.object1.collisionName == "windows" ) then
                event.object1:removeSelf()
            else
                event.object2:removeSelf()
            end

        end
    end

end
end
```

Functions - Touches

```
-- This method is called to handle a touch event.
-- In the main menu, if you touch the left side of the screen, you'll pick Bill Gates
-- In the main menu, if you touch the right side of the screen, you'll pick Steve Jobs
function buttonHandler( event )

    -- If we touched the screen then released. Ended the Touch.
    if( event.phase == "ended" ) then

        if( event.y > 250 ) then -- Make sure we touched below the game title

            if( event.x <= 320 ) then   -- Left half of the screen picks Bill Gates

                player = "gates"
                startGame() -- Begin the game

            else                        -- Right half of the screen picks Steve Jobs

                player = "jobs"
                startGame() -- Begin the game

            end

        end

    end
end

-- Show the main menu
function showMenu()

    menu_img = display.newImage( "assets/menu.png" , true) -- display the menu background image
    Runtime:addEventListener( "touch", buttonHandler ) -- register to receive touch events

end
```

Functions - Physics

```
-- Starts the Physics Engine and Sets the Gravity for the Game
function startPhysics()
    -- Start the physics simulation
    physics.start()

    physics.setScale( 60 ) -- resolution of physics simulation
    physics.setGravity( 0, 9.8 ) -- set gravity to point down
end
```

Functions - Game

```
-- This method will start a new game.
function startGame( )

    -- Clean up menu
    Runtime:removeEventListener( "touch", buttonHandler )
    menu_img:removeSelf()
    menu_img = nil

    -- Load the game background
    background_img = display.newImage( "assets/background.png", true)

    -- Initialize the Display Group to keep our icons in one place
    gamePieceGroup = display.newGroup()

    -- Start the physics engine
    startPhysics()

    -- Create the player object
    createPlayer()

    -- Create the on screen text objects
    createText()

    -- Create the game floor
    createFloor()

    -- Initialize Game Status
    gameOver = false
    timeLeft = 30
    gameScore = 0

    -- Register to detect collisions
    Runtime:addEventListener( "collision", onCollision )

    -- Register to update the game on every frame
    Runtime:addEventListener( "enterFrame" , updateGame )

    -- Start updating the game timer every second
    updateGameTimer()

    -- Start spawning game pieces every second
    spawnGamePiece()
end
```

Functions - Game

```
-- This method is called when the timer runs out to show the final score
function showFinalScore()

    -- Clean up some parts of the current game

    Runtime:removeEventListener( "collision", onCollision ) -- Stop listening to collision events
    Runtime:removeEventListener( "enterFrame" , updateGame ) -- Stop updating the game every frame

    -- Remove the onscreen text objects
    scoreText:removeSelf()
    scoreText = nil

    timeText:removeSelf()
    timeText = nil

    -- Initialize and show the final score in the middle of the screen
    finalScoreText = display.newText("Score: " .. gameScore, 0, 0, native.systemFont, 100)
    finalScoreText:setTextColor(255, 255, 255, 200)

    finalScoreText:setReferencePoint(display.CenterReferencePoint)
    finalScoreText.x, finalScoreText.y = 320, 480
end
```

Functions - Game

```
-- This method is called to finish cleaning up the game and go back to the main menu.
function endGame( )

    -- Finish cleaning up game
    background_img:removeSelf()
    background_img = nil

    player_img:removeSelf()
    player_img = nil

    finalScoreText:removeSelf()
    finalScoreText = nil

    gamePieceGroup:removeSelf()
    gamePieceGroup = nil

    timeLeft = 0
    gameScore = 0
    gameOver = false

    floor:removeSelf()
    floor = nil

    showMenu()
end
```

Start the game!

```
---------- LAUNCH THE MAIN MENU --------------

showMenu()
```

Questions?