



Introduction to Mobile Application - Android Programming with Corona

Bo-Jhang Ho
CS Department, UCLA
bojhang@cs.ucla.edu
Aug 01-02, 2018



I think you're ready !!



Before capturing Pokemon, what you need to do:



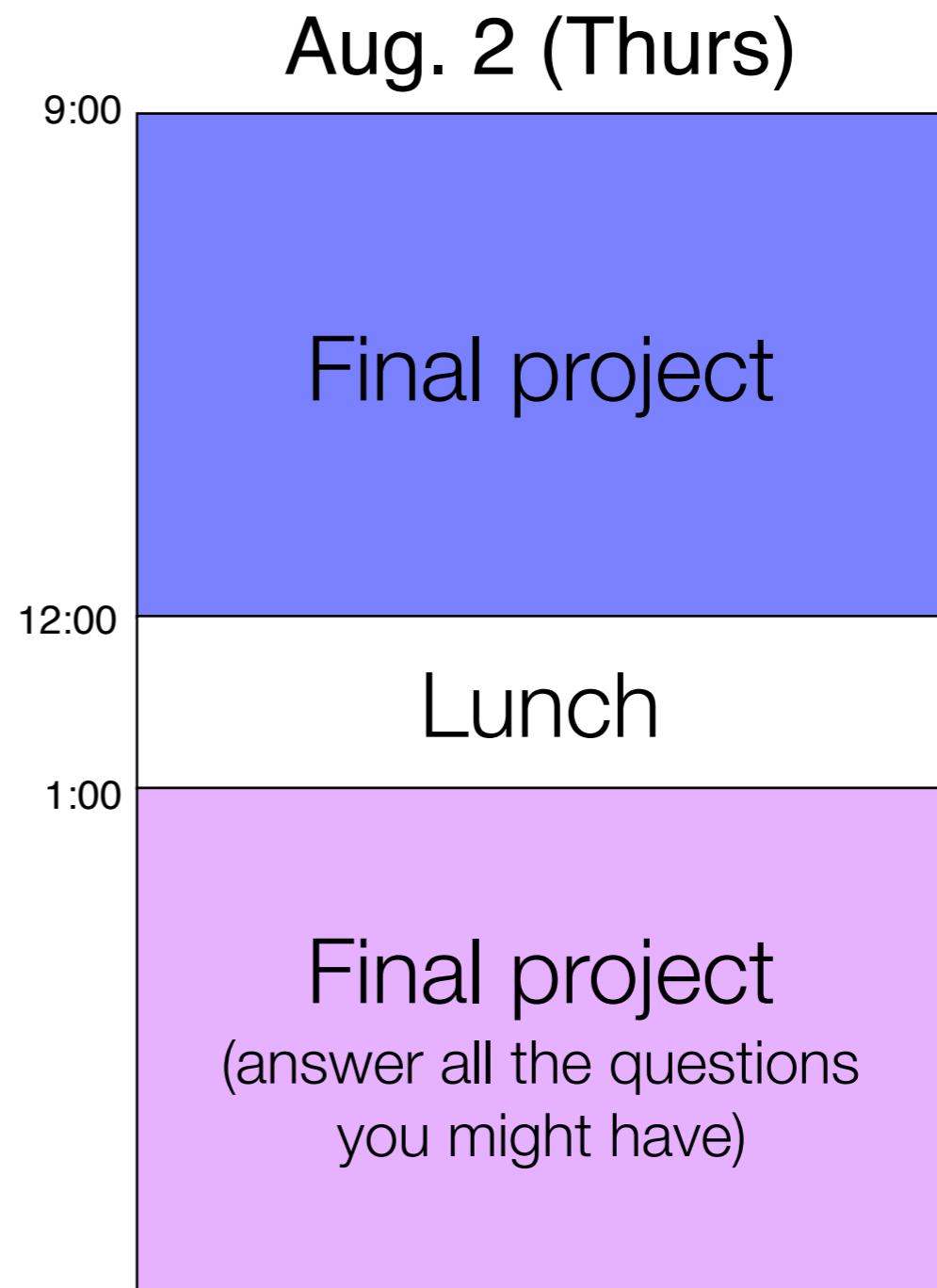
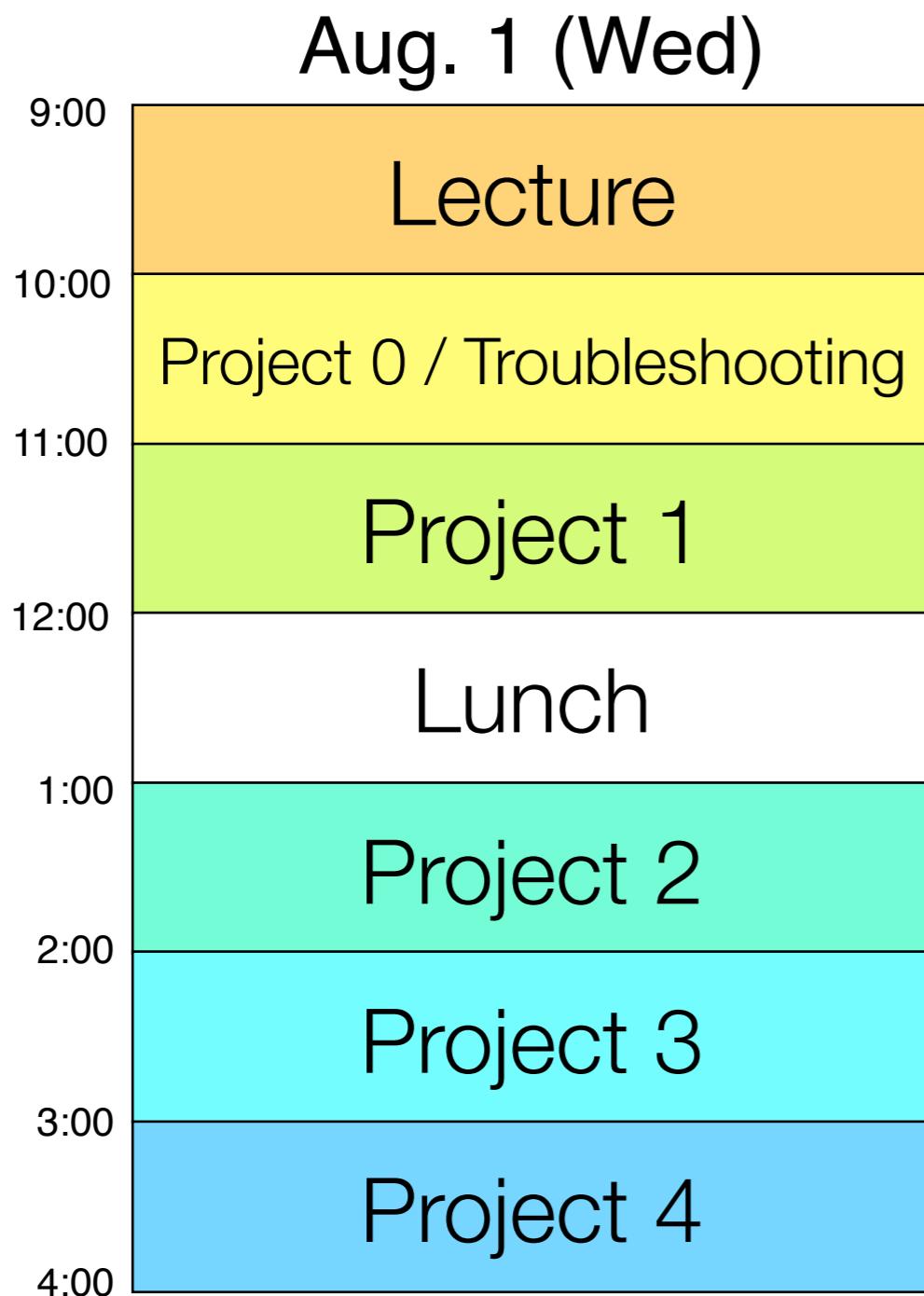
Corona SDK

1. Download this

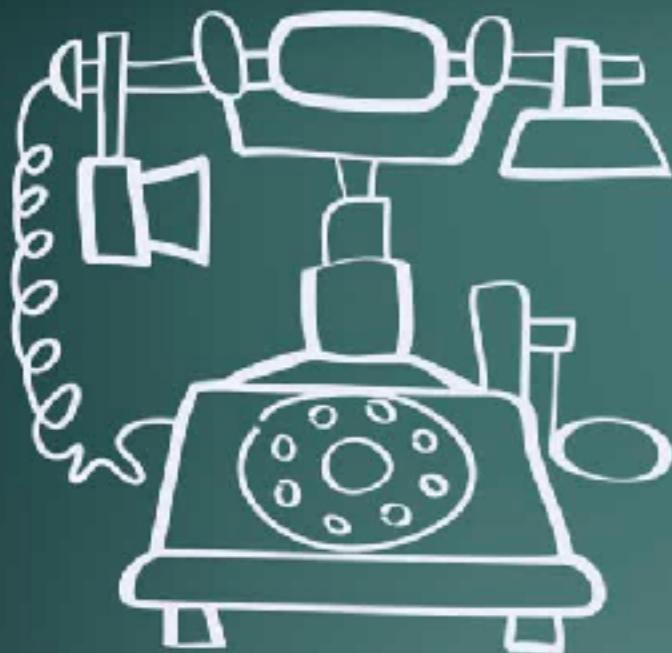


2. And download this

Agenda



Is a smartphone a phone?



Difference between a computer and a mobile phone nowadays??



Hi, I'm a PC!



I can do almost whatever you can do!

Difference between a computer and a mobile phone nowadays??



Faster computation

I can use WiFi to access Internet!

Huh?

Stationary

What's that?

What?

What?

Stop...



Decent computation speed

I can use it too!

I can use LTE

I can move almost anywhere

I have a lot of sensors!

I can know where I am!

I can know how I was moved

I can ...

Difference between a computer and a mobile phone nowadays??



I can only play Pokemon...



I can play Pokemon Go!





BTW, I have brothers & sisters!
Smart devices around us



Capability of modern smartphones

- Smartphones (and other smart devices) are everywhere!
 - The number of global smartphone users has surpassed 2 billion people!
- Multiple processor cores
- Multiple interfaces or ways to communicate with humans
 - visual, audio, haptic, blinking, ...
- Many built-in sensors
 - Motion sensors, orientation, location, temperature, camera, microphone
- Radios for communication
 - Bluetooth, WiFi access points, NFC, ...
- Cameras + powerful CPU / GPU to enable augmented reality
 - ARKit, Deep learning!



Mobile applications

- Why are smartphones so popular? What apps do you use?



Real case study

- Social
- Games
- Location Service
- Learning
- Health
- Calendar
- Financial
- Productive



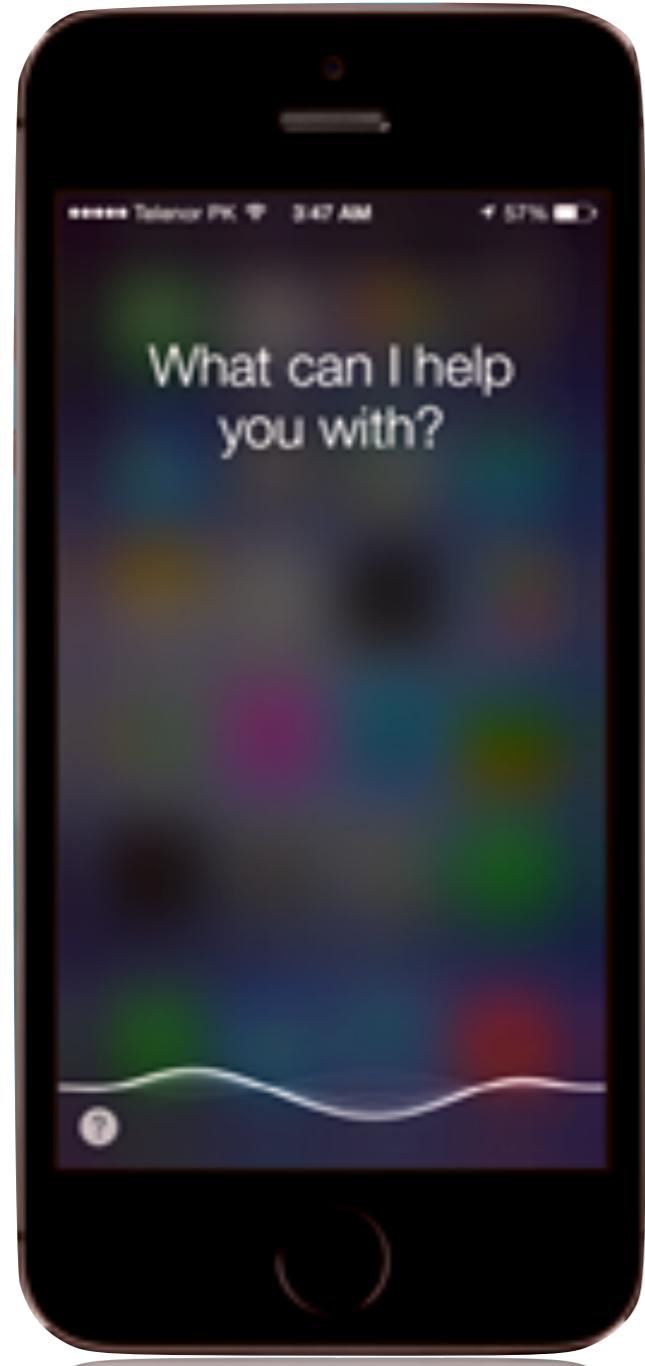
Popular mobile app development model



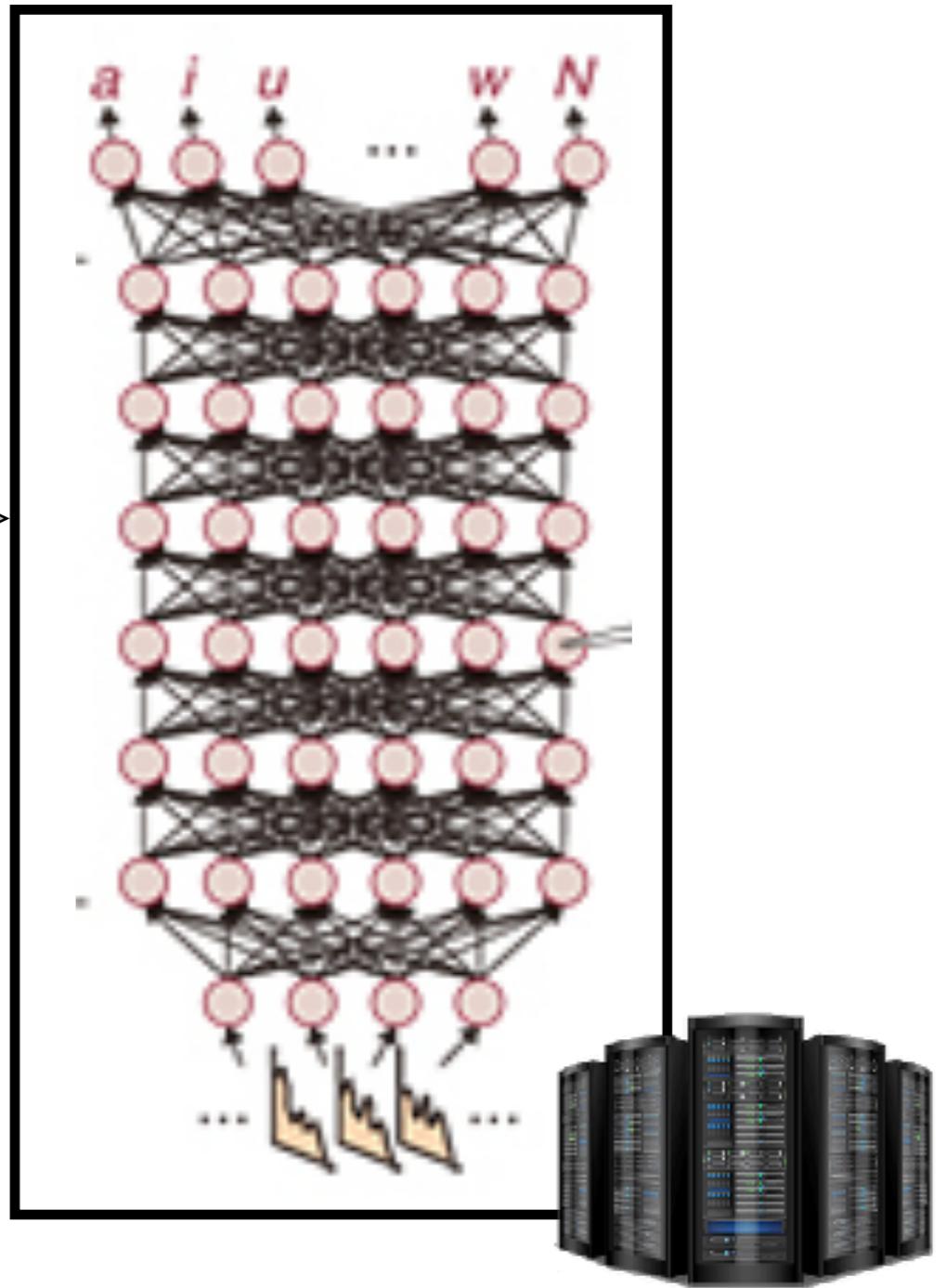
Popular mobile app development model



Popular mobile app development model



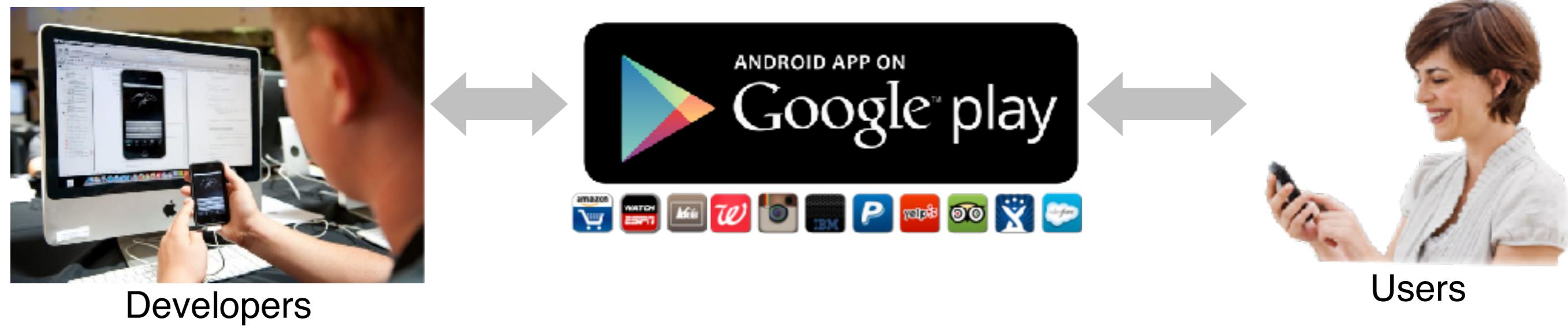
Audio data
→
←
Text-based command



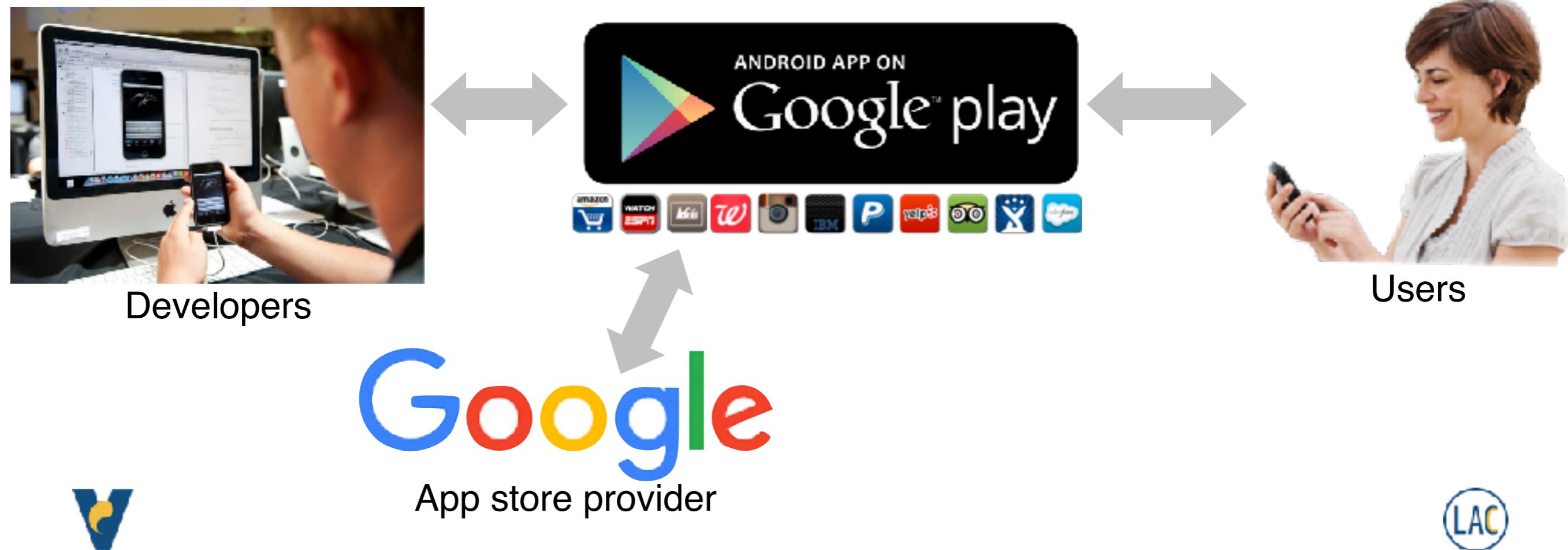
Mobile development ecosystem



Mobile development ecosystem



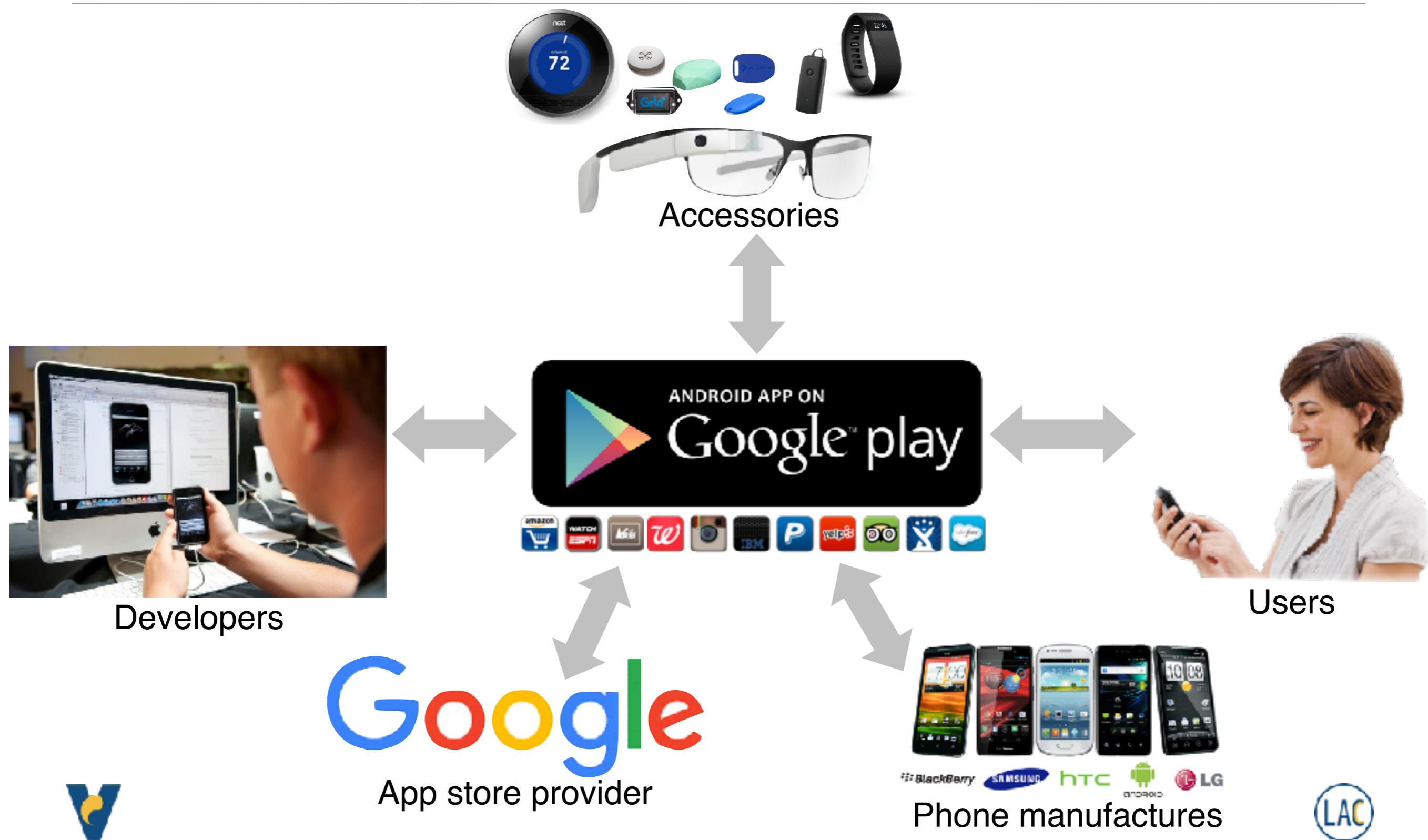
Mobile development ecosystem



Mobile development ecosystem



Mobile development ecosystem



Mobile development ecosystem

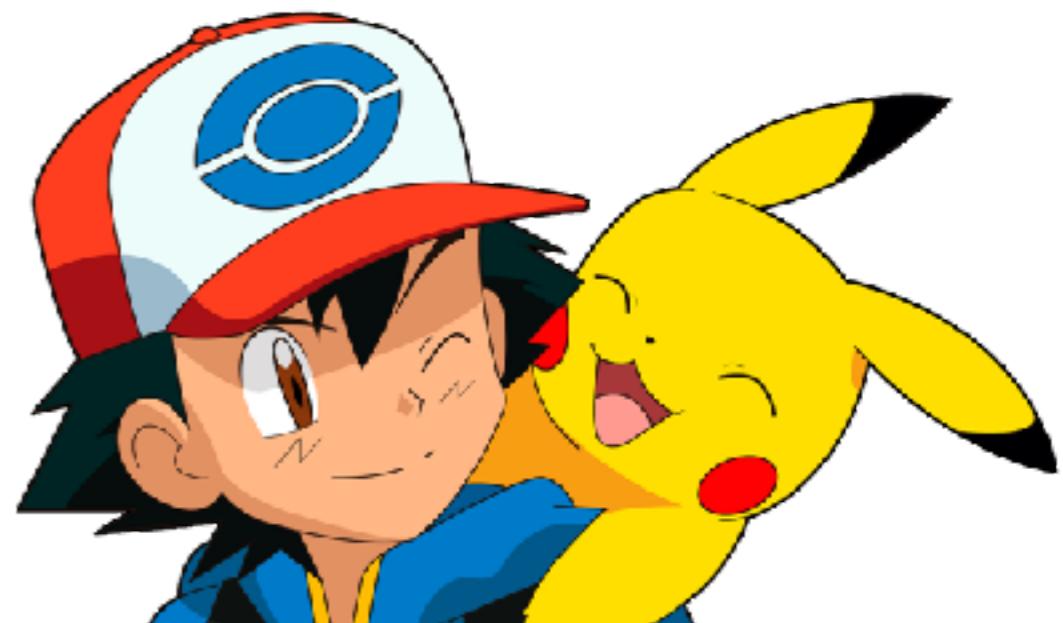


Mobile development ecosystem

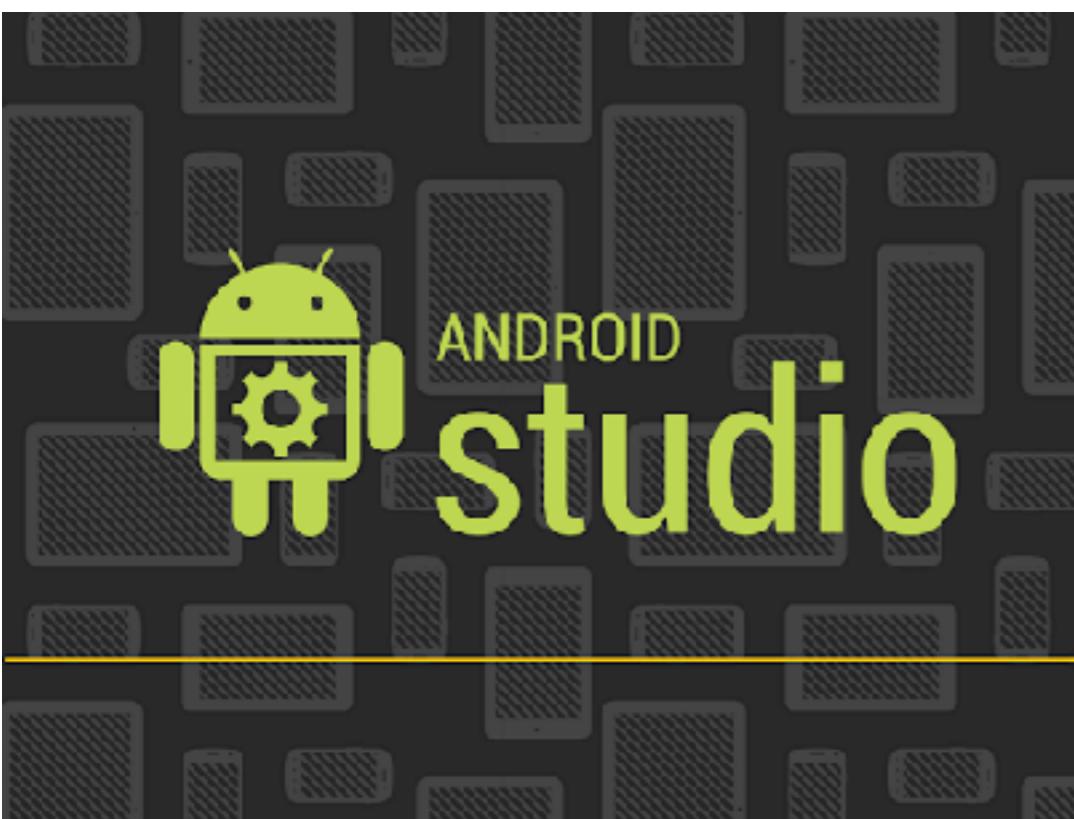


**Hold your Poke balls
and identify your Pokemon!**

Android app development using Corona SDK



Popular mobile development platforms



Java

Kotlin 



Objective-C

Swift 





Popular mobile development platforms

The screenshot shows the Android Studio interface with the project 'streamoplayer' open. The left sidebar displays the project structure, including the 'src' directory which contains the package 'co.infinum.streamoplayer' with classes like MainActivity, StreamActivity, StreamApp, and UserSettingsActivity. The main editor window shows the 'MainActivity.java' file with its code:

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import co.infinum.streamoplayer.utils.JSONParser;

public class MainActivity extends Activity {

    private static String NEW_DEVICE_URL_EXTENSION = "/api/new_device";
    private static final String TAG_RESULT = "result";
    private static final String TAG_MESSAGE = "message";
    private static final String PARAMETER_NAME = "name";
    private static final int RESULT_SETTINGS = 1;
    private static final int RESULT_STREAM = 2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button submitButton = (Button) findViewById(R.id.submitButton);
        Button streamButton = (Button) findViewById(R.id.streamButton);
        showUserSettings();

        submitButton.setOnClickListener(v -> {
            registerDevice();
            // new RegisterDevice().execute(newDeviceURL(), null, null);
        });

        streamButton.setOnClickListener(v -> {
            Intent streamIntent = new Intent(getApplicationContext(), StreamActivity.class);
            startActivityForResult(streamIntent, RESULT_STREAM);
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent;
        switch (item.getItemId()) {

            case R.id.menu_settings:
                intent = new Intent(this, UserSettingsActivity.class);
                startActivityForResult(intent, RESULT_SETTINGS);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```



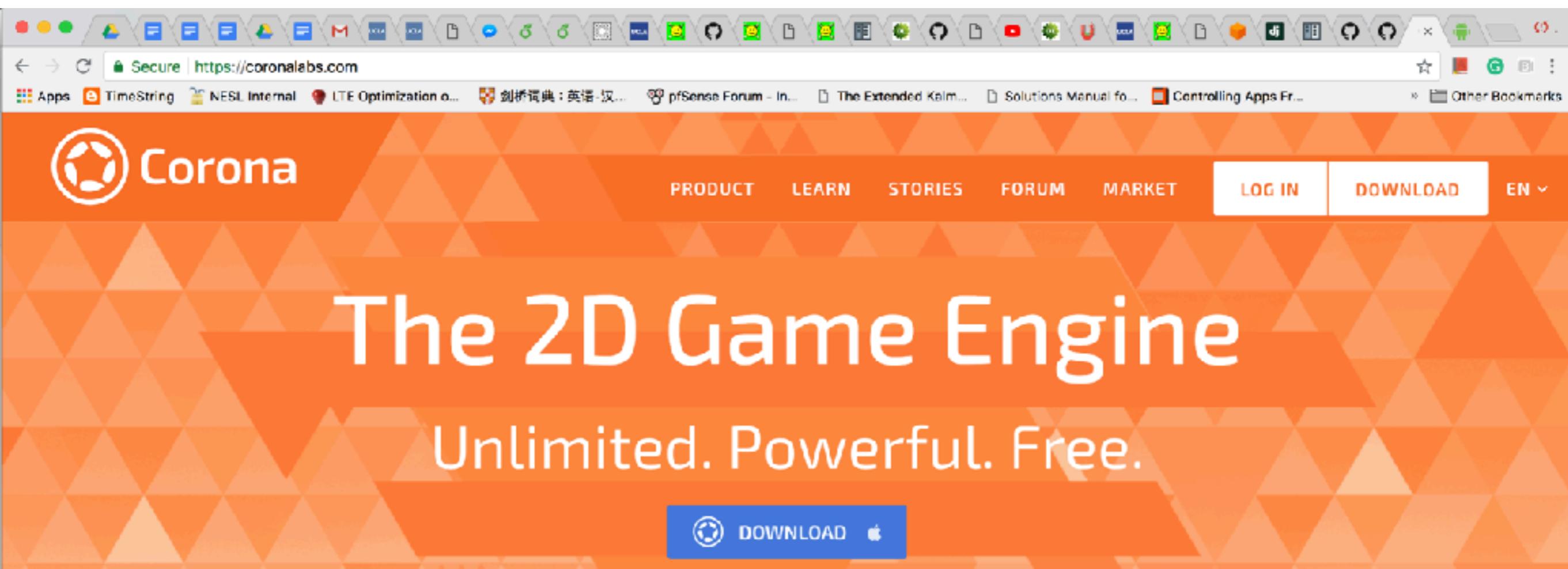
Popular mobile development platforms

The screenshot shows the Xcode interface with three main panes:

- Left Pane:** Displays the Swift code for the `setupHero` function. The code handles Blimp Control, Scene Configuration, balloon lighting, per-pixel collisions, and initializes turbulent field forces.
- Middle Pane:** Shows the game's preview window titled "Balloons". It features a blue sky, a green ground with red and white striped tents, and several colorful balloons (red, green, blue, yellow) floating. A large orange blimp is visible in the background.
- Bottom Right Pane:** Displays a graph of a sine wave with the equation $y = 80 \cdot \sin(x)$. The x-axis ranges from -30 to 30 seconds, and the y-axis ranges from -50 to 50.

A lot of code!!

Let's why we choose Corona SDK



- Plus, you can compile on both iOS and Android platforms!
- No need to deal with the complex library and build system
- Provide a bunch of common libraries



Recall Android Studio

The screenshot shows the Android Studio IDE interface. The top navigation bar includes icons for file operations like Open, Save, and Build, followed by the project name "streamoplayer". Below the navigation bar is the toolbar with various icons for project management.

The left side features the Project Structure sidebar, which displays the project's directory tree. It includes sections for Project, src, and res. The src folder contains packages like co.infinum.streamoplayer and sub-packages like utils, which contain MainActivity, StreamActivity, StreamApp, and UserSettingsActivity. The res folder contains assets, bin, gen, libs, and layout folders, with layout containing activity_main.xml, activity_stream.xml, and activity_user_settings.xml files.

The main content area shows the code editor with the file "MainActivity.java" open. The code implements the Activity interface, handling onCreate(), onCreateOptionsMenu(), and onOptionsItemSelected() methods. It uses findViewById() to get references to submit and stream buttons, sets their click listeners, and starts activities for result codes RESULT_SETTINGS and RESULT_STREAM. It also handles options menu inflation and item selection.

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import co.infinum.streamoplayer.utils.JSONParser;

public class MainActivity extends Activity {

    private static String NEW_DEVICE_URL_EXTENSION = "/api/new_device";
    private static final String TAG_RESULT = "result";
    private static final String TAG_MESSAGE = "message";
    private static final String PARAMETER_NAME = "name";
    private static final int RESULT_SETTINGS = 1;
    private static final int RESULT_STREAM = 2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button submitButton = (Button) findViewById(R.id.submitButton);
        Button streamButton = (Button) findViewById(R.id.streamButton);
        showUserSettings();

        submitButton.setOnClickListener(v -> {
            registerDevice();
            // new RegisterDevice().execute(newDeviceURL(), null, null);
        });

        streamButton.setOnClickListener(v -> {
            Intent streamIntent = new Intent(getApplicationContext(), StreamActivity.class);
            startActivityForResult(streamIntent, RESULT_STREAM);
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent;
        switch (item.getItemId()) {
            case R.id.menu_settings:
                intent = new Intent(this, UserSettingsActivity.class);
                startActivityForResult(intent, RESULT_SETTINGS);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Recall Android Studio

Run

1. Compile
2. Transfer file to the phone



The screenshot displays the Android Studio interface. On the left, the Project Structure sidebar shows the project's directory tree, including the main source code folder 'streamoplayer' containing 'MainActivity.java', 'StreamActivity.java', 'StreamApp.java', and 'UserSettingsActivity.java'. It also lists resources like 'activity_main.xml', 'activity_stream.xml', and 'activity_user_settings.xml' under 'res/layout'. Other files like 'AndroidManifest.xml', 'build.gradle', and 'readme.txt' are visible in the 'src' and root directories. The right side of the screen shows the code editor with the 'MainActivity.java' file open. The code implements logic for handling button clicks and menu options, particularly for streaming and user settings. A red arrow points from the word 'Run' at the top to the play button icon in the toolbar above the code editor.

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import co.infinum.streamoplayer.utils.JSONParser;

public class MainActivity extends Activity {

    private static String NEW_DEVICE_URL_EXTENSION = "/api/new_device";
    private static final String TAG_RESULT = "result";
    private static final String TAG_MESSAGE = "message";
    private static final String PARAMETER_NAME = "name";
    private static final int RESULT_SETTINGS = 1;
    private static final int RESULT_STREAM = 2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button submitButton = (Button) findViewById(R.id.submitButton);
        Button streamButton = (Button) findViewById(R.id.streamButton);
        showUserSettings();

        submitButton.setOnClickListener((v) -> {
            registerDevice();
            // new RegisterDevice().execute(newDeviceURL(), null, null);
        });

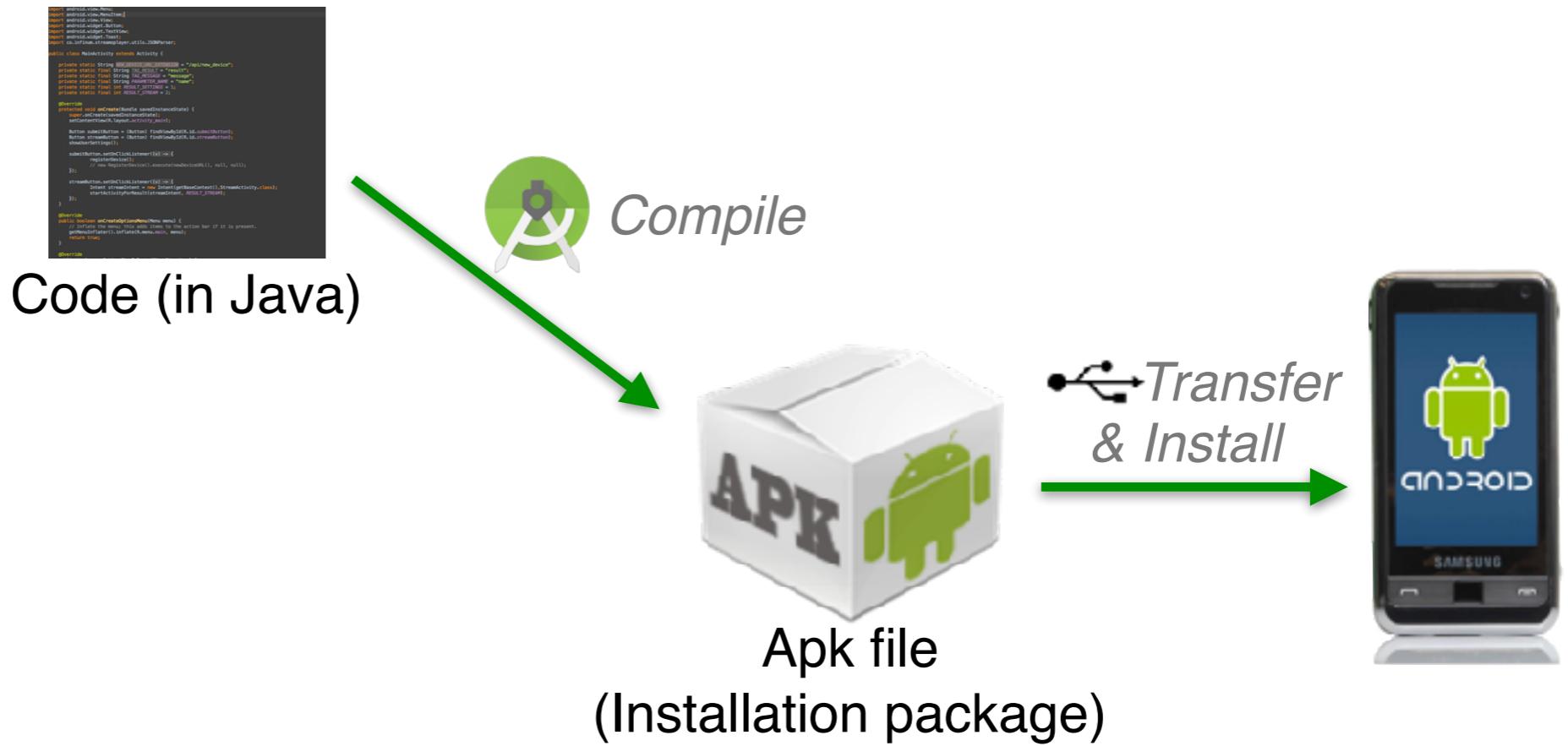
        streamButton.setOnClickListener((v) -> {
            Intent streamIntent = new Intent(getApplicationContext(), StreamActivity.class);
            startActivityForResult(streamIntent, RESULT_STREAM);
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent;
        switch (item.getItemId()) {

            case R.id.menu_settings:
                intent = new Intent(this, UserSettingsActivity.class);
                startActivityForResult(intent, RESULT_SETTINGS);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

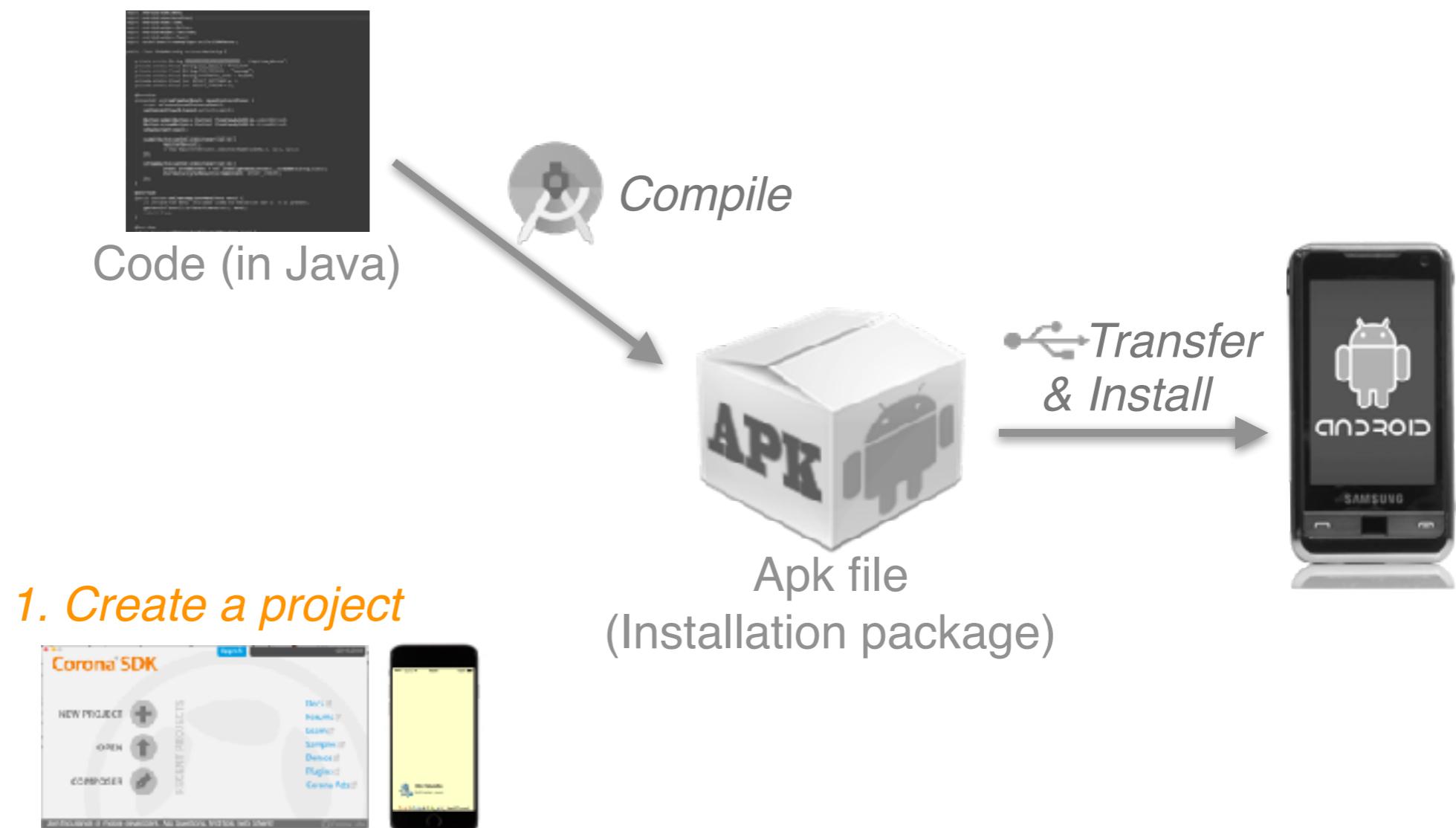
Compilation path of Android





Corona SDK

Working flow in Corona SDK



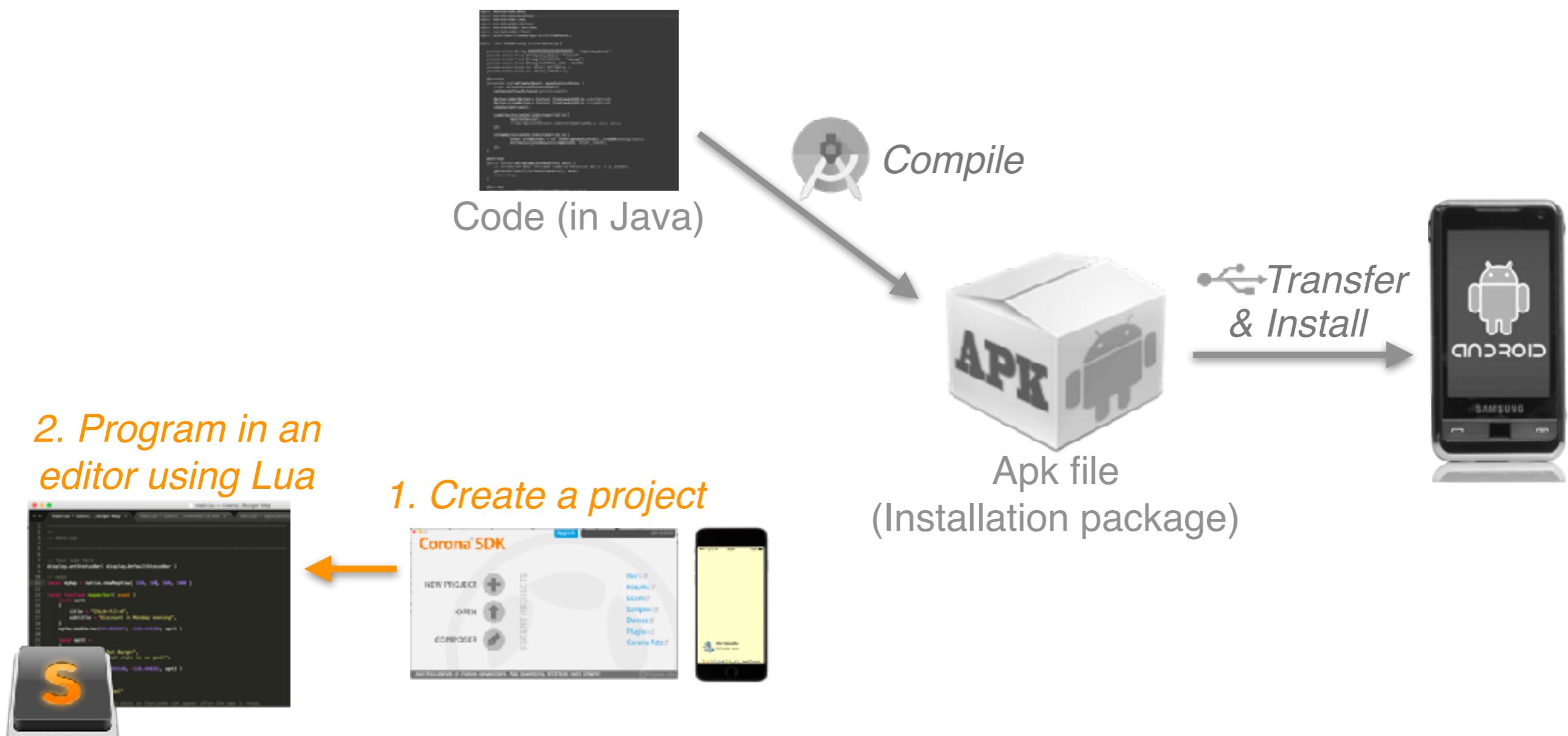
www.variability.org





Corona SDK

Working flow in Corona SDK



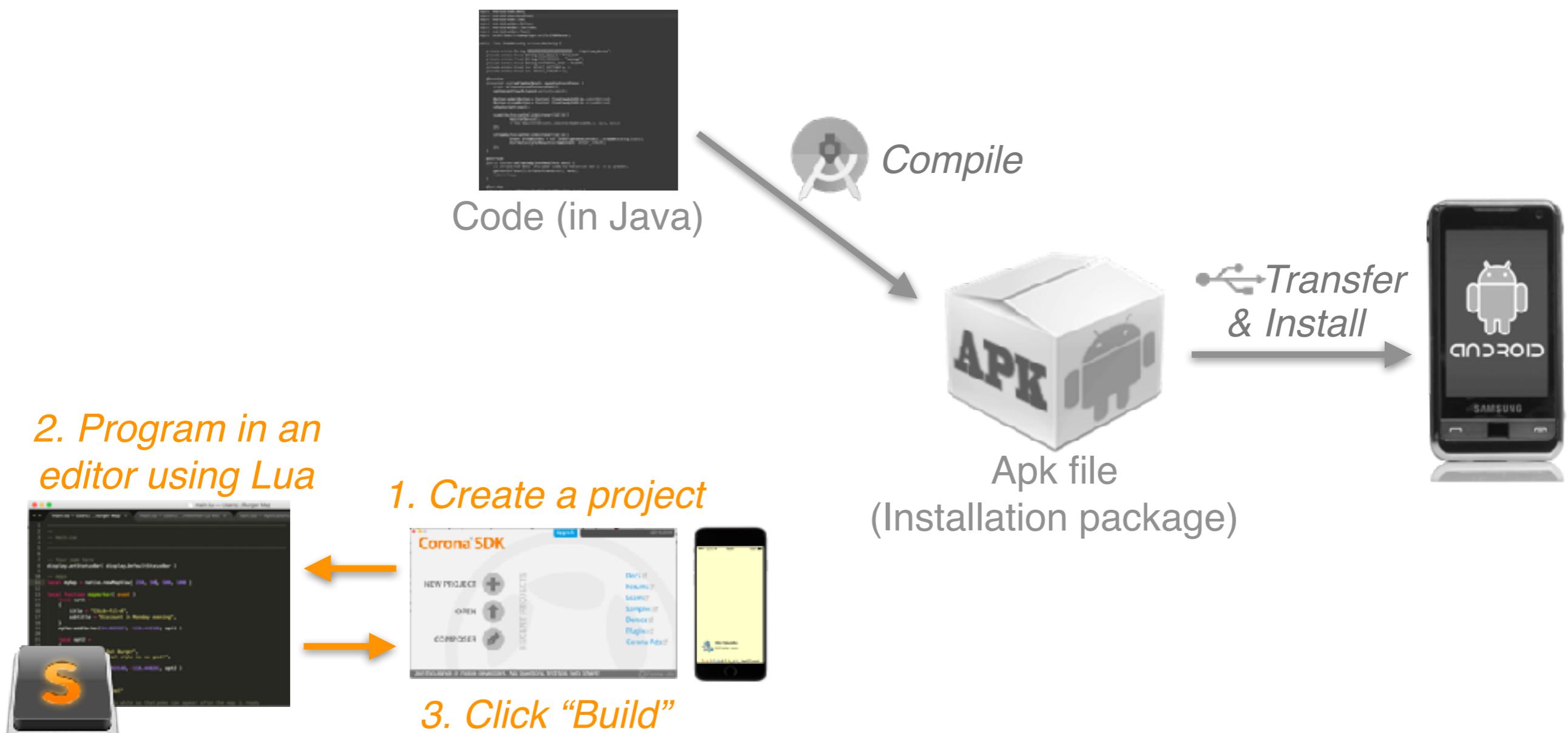
www.variability.org





Corona SDK

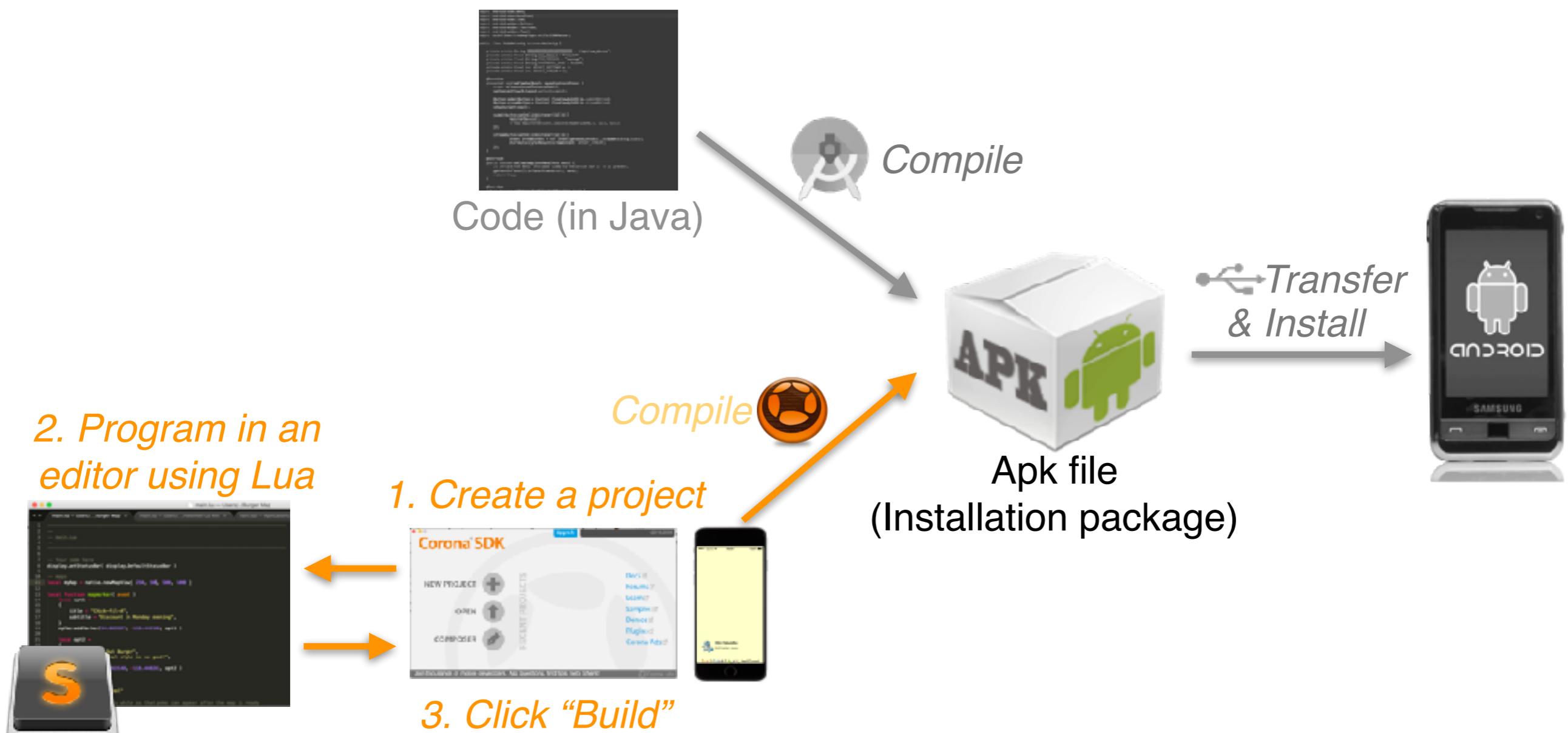
Working flow in Corona SDK





Corona SDK

Working flow in Corona SDK



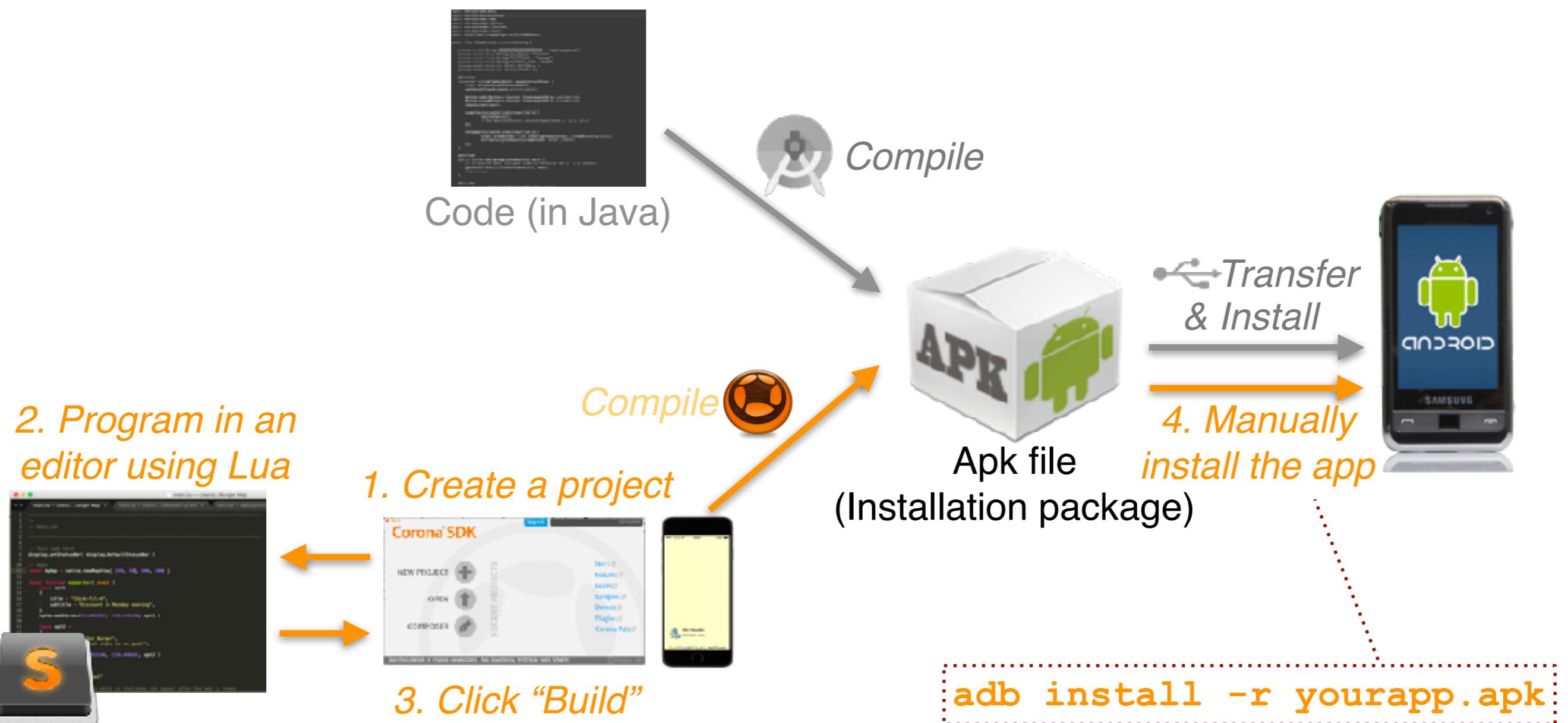
www.variability.org





Corona SDK

Working flow in Corona SDK



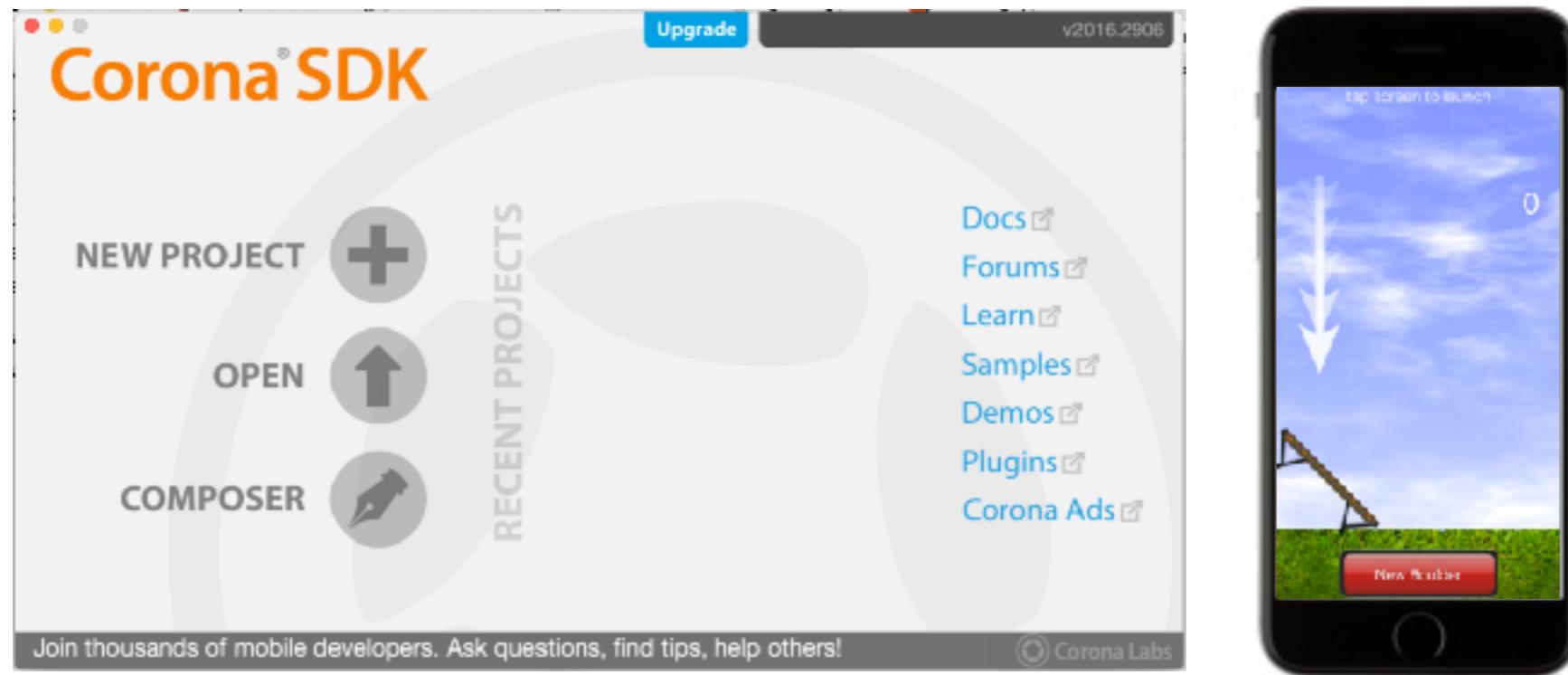
Episode 0 - There is a Gym...

Warm up



Project 00:

Run a sample program using Corona



- Checkpoint 1: Be able to open Corona SDK
- Checkpoint 2: Build the app
- Checkpoint 3: Run the app on a phone

Introduction to Lua - general conventions

- Tip 1: Use variables
 - When the variable appears in your code the very first time, put keyword `local` in front of it (Don't ask why!)
 - A variable can be a number, a boolean, a string a function, a table (?!)
 - Examples:

```
local a = 34
local b = "This is a string"
local c = true

local d = function(arg1, arg2)
    return arg1 + arg2
end

local e = {1, 2, "three", 4.1}
```

- Tip 2: Don't use **reserved** keywords as variable names!

| | | | | | | |
|-----|----------|--------|------|--------|-------|-------|
| and | break | do | else | elseif | end | false |
| for | function | if | in | local | nil | not |
| or | repeat | return | then | true | until | while |

Can you tell me more about a table?



```
local d = {1, 2, "three", 4.0}
```

is equivalent to

```
local d = {}  
d[1] = 1  
d[2] = 2  
d[3] = "three"  
d[4] = 4.0
```

| Field (index) | Value |
|---------------|-------|
| 1 | 1 |
| 2 | 2 |
| 3 | three |
| 4 | 4.0 |

- The field can be a string, too

```
local d = {}  
d.name = "Bo-Jhang"  
d.occupation = "Student"  
d.city = "Los Angeles"
```

=

```
local d = {}  
d["name"] = "Bo-Jhang"  
d["occupation"] = "Student"  
d.city = "Los Angeles"
```

=

| Field | Value |
|------------|-------------|
| name | Bo-Jhang |
| occupation | Student |
| city | Los Angeles |



Expressions

- Tip 3: Use arithmetic operators

```
local a = 34
local b = 1.5 * 2.0
local c = (a + 6) * b
print(c)
```

| | | | |
|---|------------------|---|------------------|
| + | (addition) | - | (subtraction) |
| * | (multiplication) | / | (division) |
| % | (modulo) | ^ | (exponentiation) |

- Tip 4: Relational operators

```
if a > 20 then
    print("apple")
end
```

```
if a < 20 then
    print("Bulbasour")
else
    print("Ivysaur")
end
```

```
if a < 20 then
    print("Eevee")
elseif a < 30 then
    print("Vaporeon")
elseif a < 40 then
    print("Jolteon")
else
    print("Flareon")
end
```

| | |
|----|----------------------------|
| == | (equal to) |
| ~= | (not equal to) |
| < | (less than) |
| > | (greater than) |
| <= | (less than or equal to) |
| >= | (greater than or equal to) |

- Tip 4.2: Logical Operators - and, or, not

```
if a < 20 or b < 4 then
    print("Delta Airline")
elseif a > 20 and b < 6 then
    print("Virgin America")
end
```



Expressions

- Tip 5: String concatenation

```
local a = "one" ← one
local b = "two" ← two
local c = a .. b ← onetwo
local d = 10 ← 10
local e = c .. d ← onetwo10
```

Two dots does the trick



- Tip 6: Repetition - using loops

- Two ways to create a loop, don't get confused!

Start End

```
for i = 1, 10 do
| print(i)
end
```

We'll get 10 numbers:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Start End Step

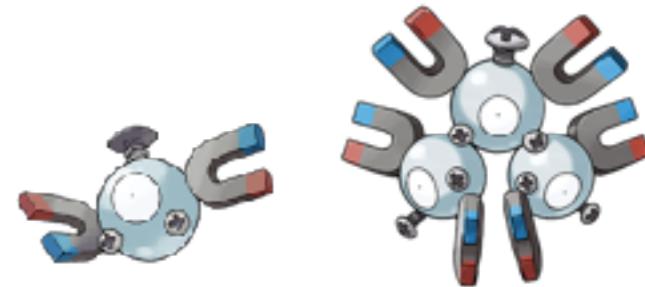
```
for i = 1, 10, 3 do
| print(i)
end
```

We'll get 4 numbers:
1, 4, 7, 10

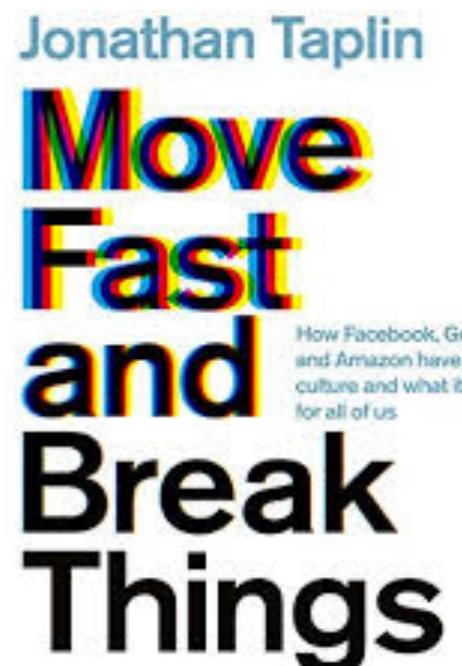
Accessing attributes of objects

- Tip 7: Sometimes we use a dot, sometimes we use a colon!
 - If you want to become a master, don't question!

```
local textMorePoke = display.newText("")  
textMorePoke.x = 420  
textMorePoke.y = 980  
textMorePoke:setFillColor(0, 0, 0)
```



- Tip 8: Take sample codes as a reference
- Tip 9: Be confident! Ask the instructor! Ask TAs!
 - Search on Google, copy and paste, modify

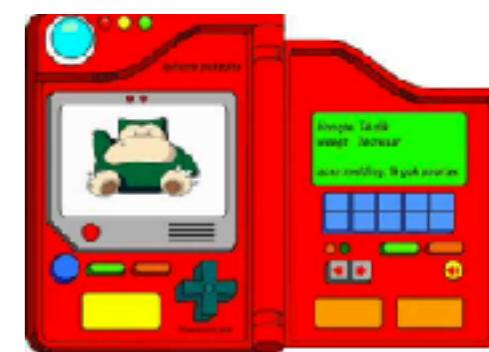


Episode 1 - Pokedex

Calculator App



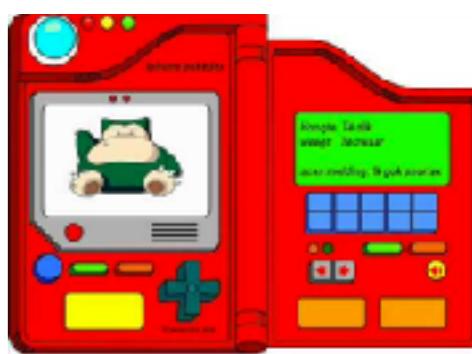
Project 01: Calculator App



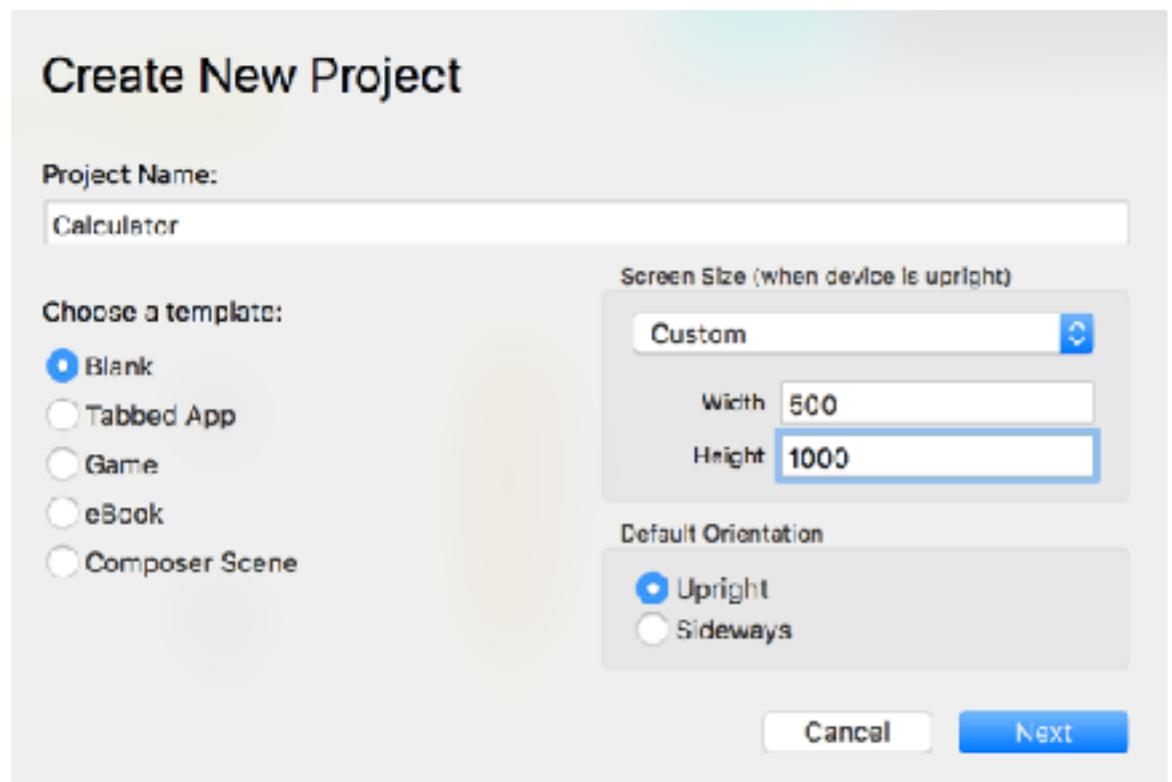
- Two blanks for users to enter numbers
- Arithmetic buttons
- Give results



Project 01: Calculator App



- Step 1: Create a new project



- Step 2: Open main.lua in your editor

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

```
main.lua
```

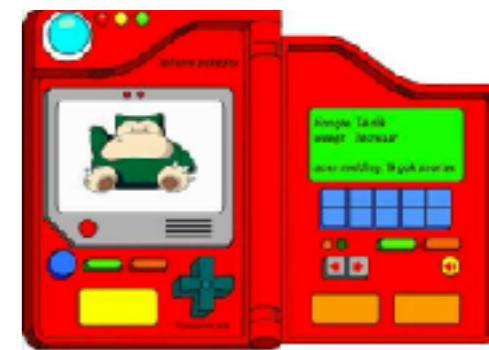
```
Lua script - 2 KB
```

```
Created 7/14/16, 6:10 PM
```

```
Modified 7/14/16, 6:48 PM
```

```
Last opened 7/14/16, 6:11 PM
```

```
Add Tags...
```



Project 01: Calculator App

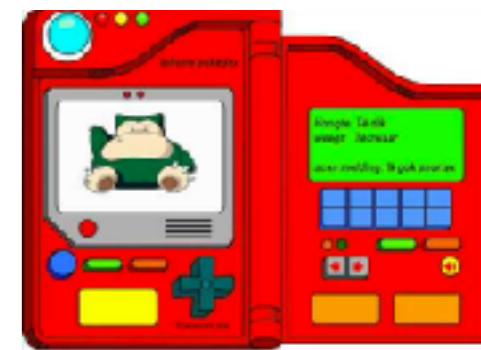
- Step 3: Always put this in the beginning of your code

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

- Step 4: Put a text input on the screen

```
local input1 = native.newTextField( 370, 80, 230, 40 )
```

center x, center y, width, height



Project 01: Calculator App

- (Optional) Step 5: Put a flat rectangle as a line

center x, center y, width, height

```
local flatRect = display.newRect(250, 240, 480, 5)  
flatRect:setFillColor(1, 1, 1)
```

| | |
r, g, b

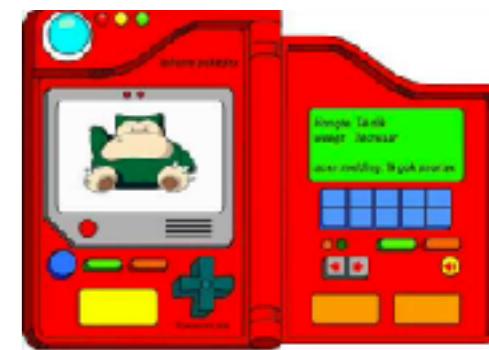
- Step 6: Put a text field to show the answer

default content, width, height, font, font size

```
local textAnswer = display.newText("???", 450, 40, native.systemFont, 32)  
textAnswer.x = 250  
textAnswer.y = 270
```

Remember it's center-x and center-y





Project 01: Calculator App

- Step 7.2: Make a button

```
local button1 = widget.newButton
{
    label = "+",
    shape = "roundedRect",
    fillColor =
    {
        default = {0.5, 0.5, 0.5},
        over = {0.6, 0.6, 0.6},
    },
    labelColor =
    {
        default = {1, 1, 1},
    },
    fontSize = 22,
    font = native.SystemFont,
    onRelease = addHandler,
    width = 45,
}

button1.x = 30
button1.y = 160
```

It's called a **callback**, which gives us an opportunity to specify the action when button is pressed (released)

We haven't seen the variable `addHandler` before, that's why we are one step ahead

- Step 7: Specify the action when the button is clicked

```
local addHandler = function(event)
    textAnswer.text = input1.text + input2.text
end
```



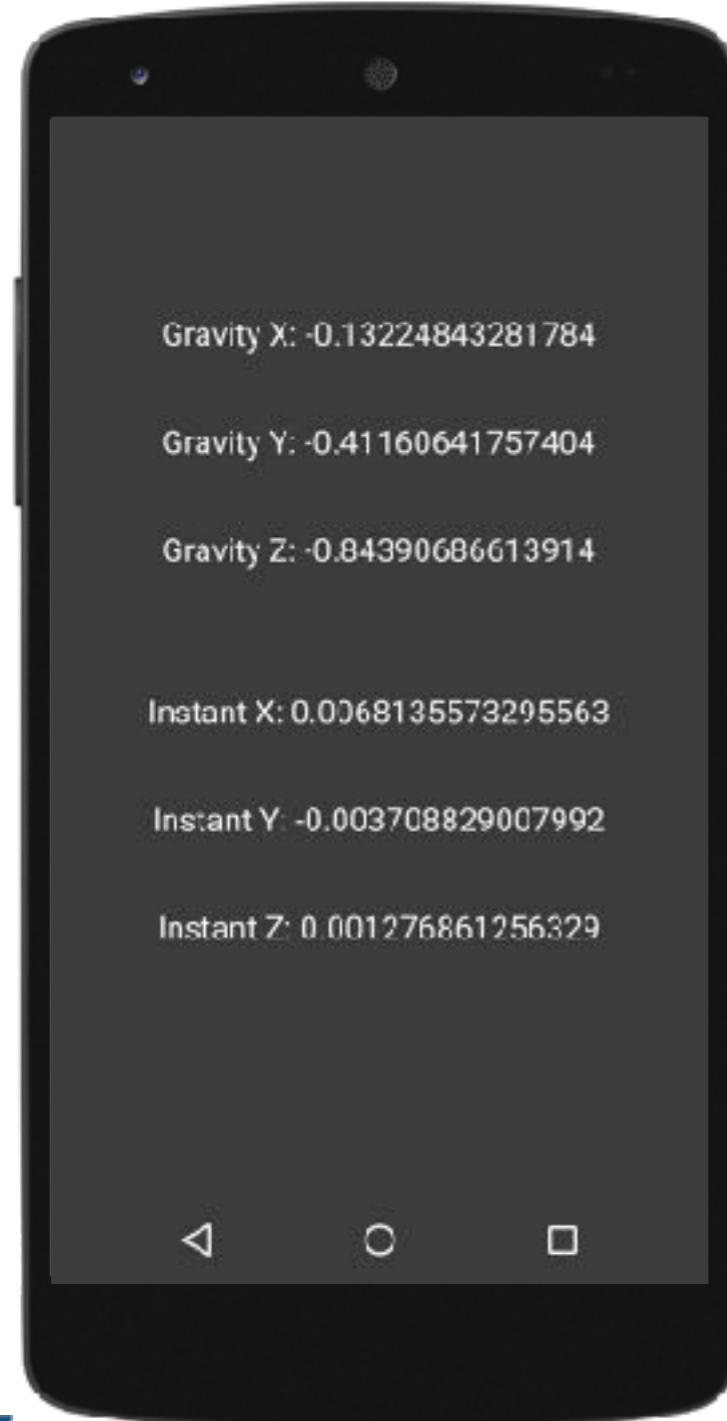
Episode 2 - How to Capture a Pokemon?

Accelerometer App





Project 02: Accelerometer App



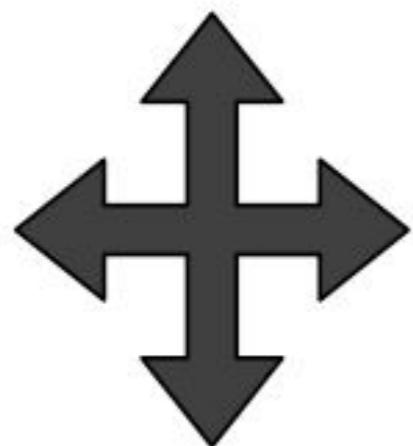
- Read values from accelerometer sensor
- Get gravity X, Y, Z and instant force X, Y, Z



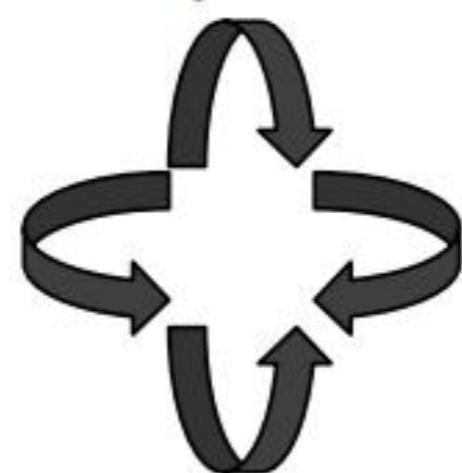


Motion sensors

Accelerometer



Gyro



Compass





Project 02: Accelerometer App

- Step 1: Get the UI done
- Step 2.2: Request accelerometer service
 - sampling rate in hertz

```
system.setAccelerometerInterval(60)  
Runtime.addEventlistener("accelerometer", onAccelerate)
```

- Step 2: What are we going to do when getting an pack of accelerometer values? Show them!

```
local onAccelerate = function(event)  
    textGravX.text = "Gravity X: " .. event.xGravity  
    textGravY.text = "Gravity Y: " .. event.yGravity  
    textGravZ.text = "Gravity Z: " .. event.zGravity  
    textInstantX.text = "Instant X: " .. event.xInstant  
    textInstantY.text = "Instant Y: " .. event.yInstant  
    textInstantZ.text = "Instant Z: " .. event.zInstant  
  
end
```



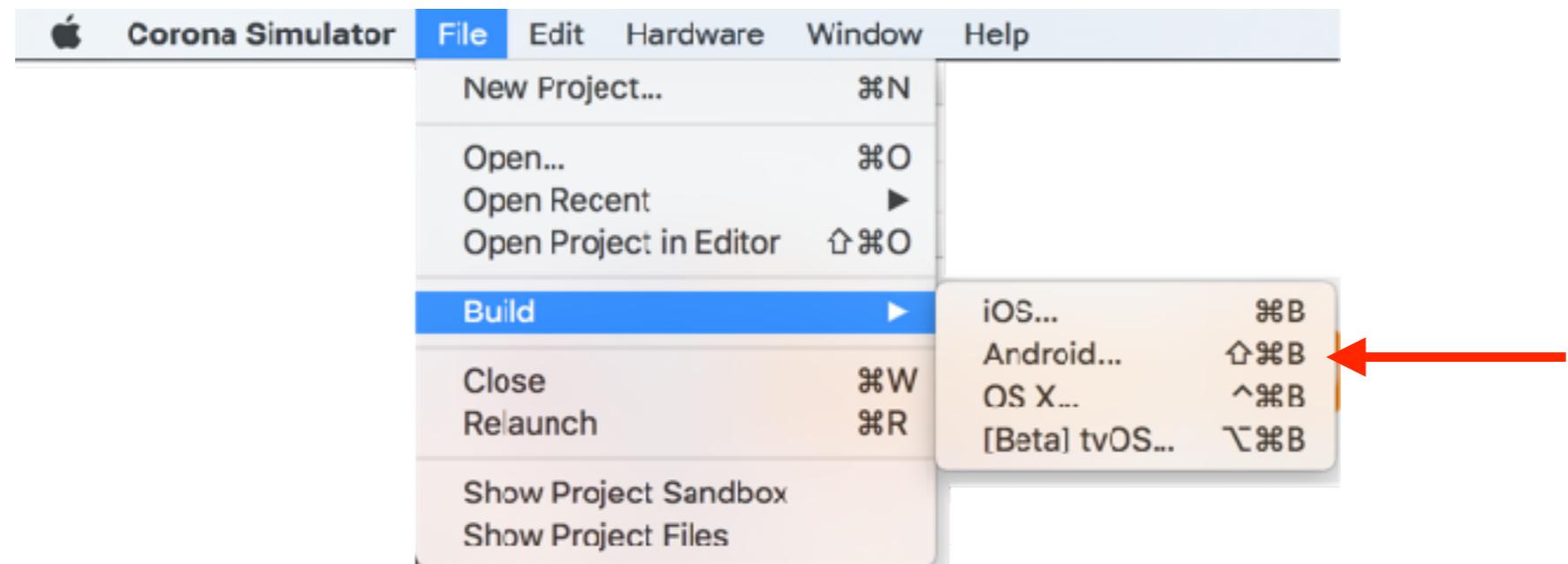
Why leaving so much space?





Project 02: Accelerometer App

- Step 3: Build the app





Project 02: Accelerometer App

- Step 4: Upload to the mobile phone
 - Connect your computer to the phone
 - Open your terminal
(in Windows, it's called *command prompt* or *cmd*)





Project 02: Accelerometer App

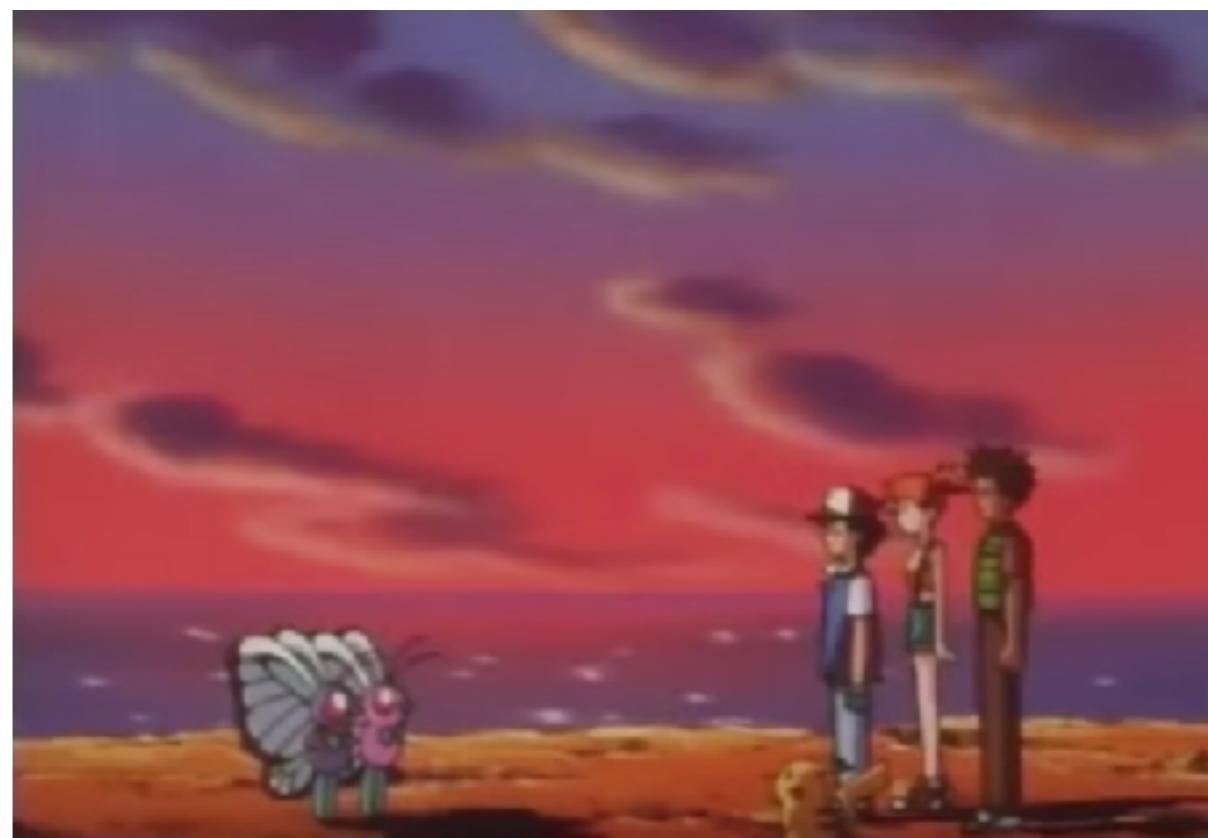
- Secret step: Shaking detection! (shhhh)

```
local onAccelerate = function(event)
    textGravX.text = "Gravity X: " .. event.xGravity
    textGravY.text = "Gravity Y: " .. event.yGravity
    textGravZ.text = "Gravity Z: " .. event.zGravity
    textInstantX.text = "Instant X: " .. event.xInstant
    textInstantY.text = "Instant Y: " .. event.yInstant
    textInstantZ.text = "Instant Z: " .. event.zInstant
    if event.isShake == true then
        toggleColor()
    end
end
```



Episode 3 - I will Come Back!

Reminder App



Project 03: Reminder App



- A reminder app which for users to take notes
- Support save/load function
- File reading/writing

Project 03: Reminder App



- Step 1: UI. Since we need to let user enter multiple lines, we have to use `newTextBox`.

center x, center y, width, height

```
input = native.newTextBox( 250, 280, 480, 480 )
input.setEditable = true
```

Project 03: Reminder App



- Step 2: Specify the file name.

```
local filePath = system.pathForFile("data.txt", system.DocumentsDirectory)
```

file name

Follow me and don't worry about it

- Step 2.1: Read the file

```
local file = io.open(filePath, "r")
if file then
    local content = file:read("*a")
    io.close(file)
    input.text = content
end
```

- Step 2.2: Write the file

```
local file = io.open(filePath, "w")
if file then
    ???
end
```

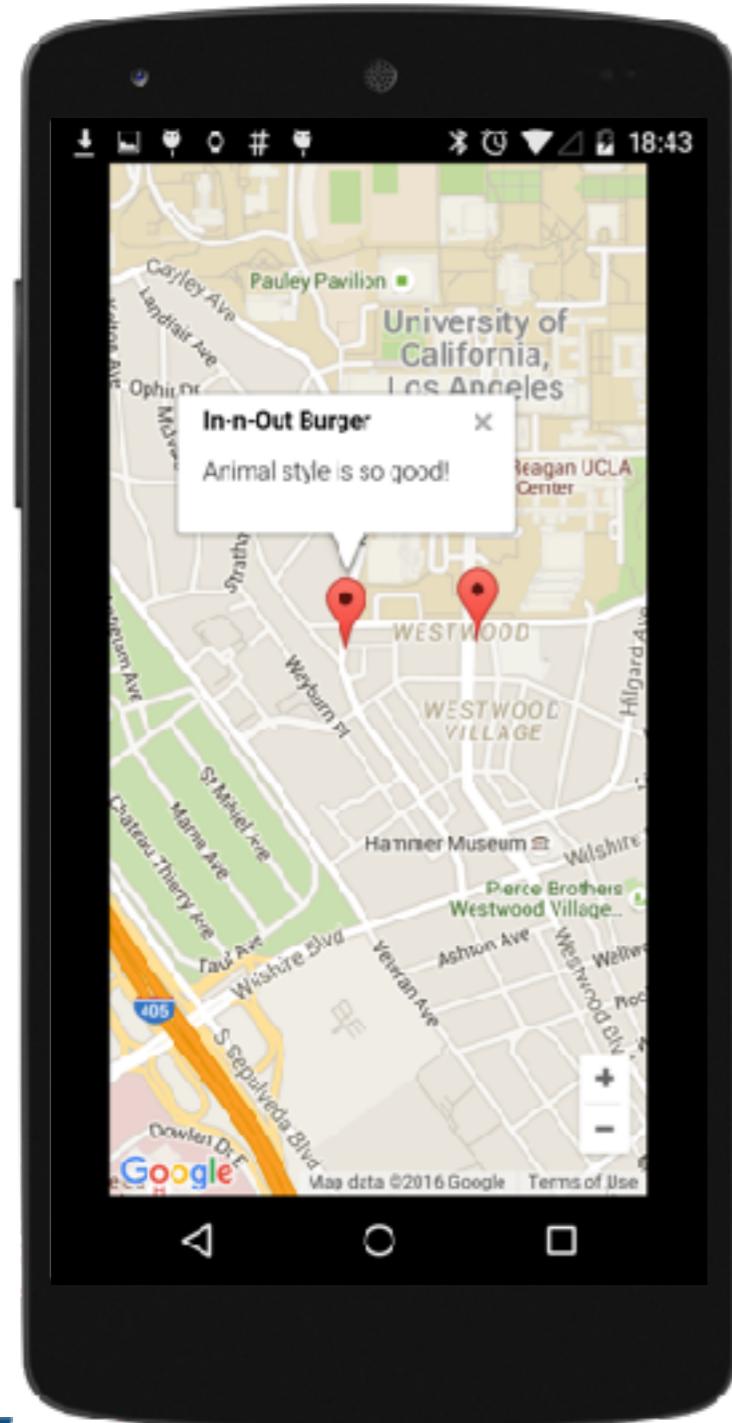
Episode 4 - Battle, I'm the Master!

Burger Map App





Project 04: Burger Map App



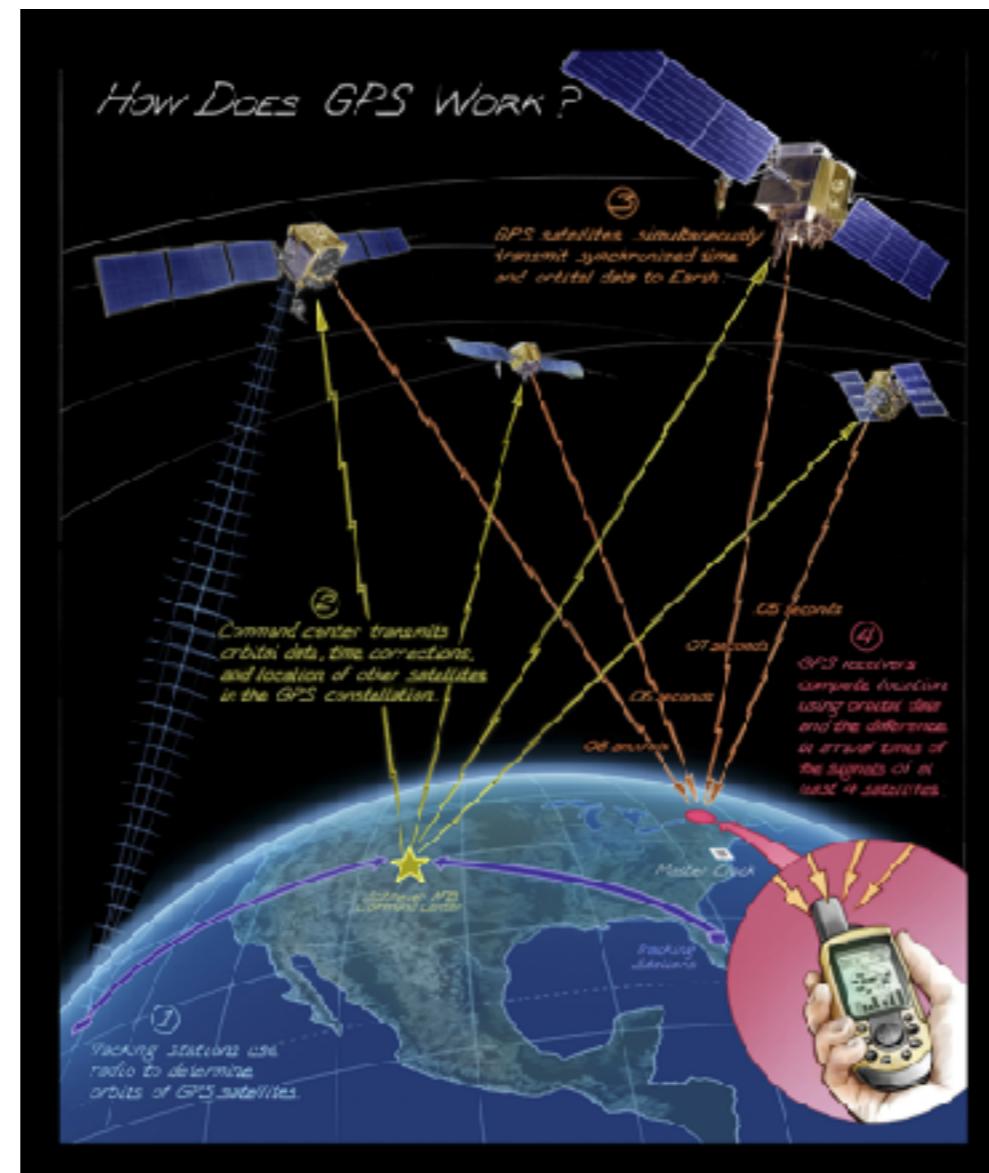
- Create a map app and select two places you want to introduce





Project 04: Burger Map App

- How do we get location? GPS! (Global Positioning System)
- The issue is that GPS is unavailable indoor
- People usually forget that WiFi plays an important role in indoor positioning





Project 04: Burger Map App

- Step 1: Open your browser and go to Google Maps, choose two places, remember their *latitude* and *longitude*
- Step 2: Create a map view

```
-- maps
local myMap = native.newMapView( 250, 500, 500, 1000 )

if myMap then
    myMap.mapType = "normal"

    -- need to delay for a while so that poke can appear after the map is ready
    timer.performWithDelay( 5000, mapmarker )
end
```





Project 04: Burger Map App

- Step 3: Put markers

```
-- maps
local myMap = native.newMapView( 250, 500, 500, 1000 )

local function mapmarker( event )
    local opt1 =
    {
        title = "Chick-fil-A",
        subtitle = "Discount in Monday evening",
    }
    myMap:addMarker(34.063327, -118.445130, opt1 )
end

if myMap then
    myMap.mapType = "normal"

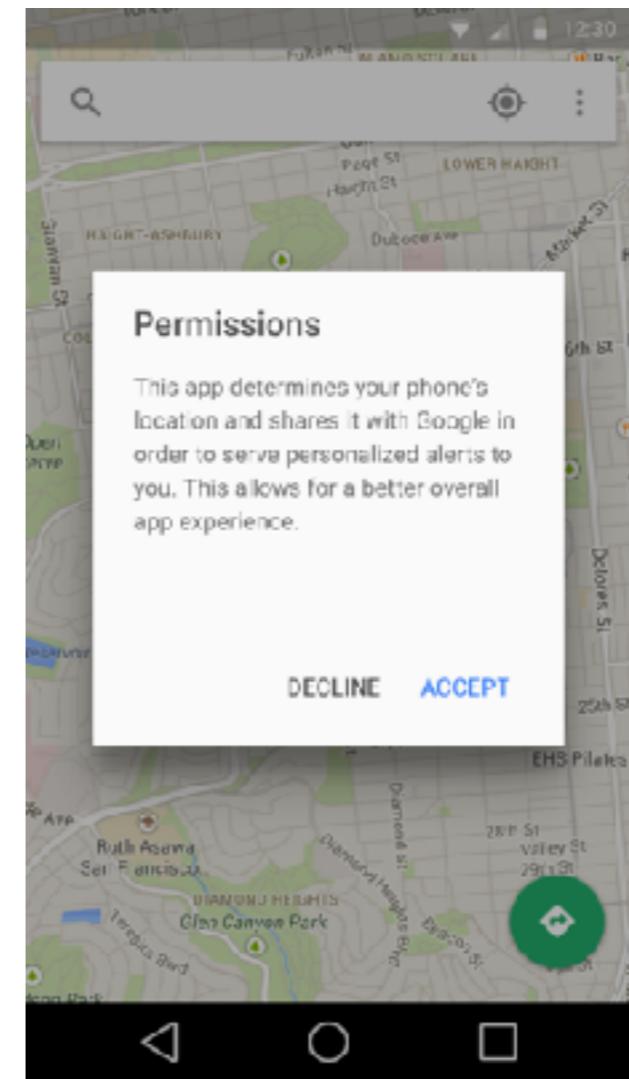
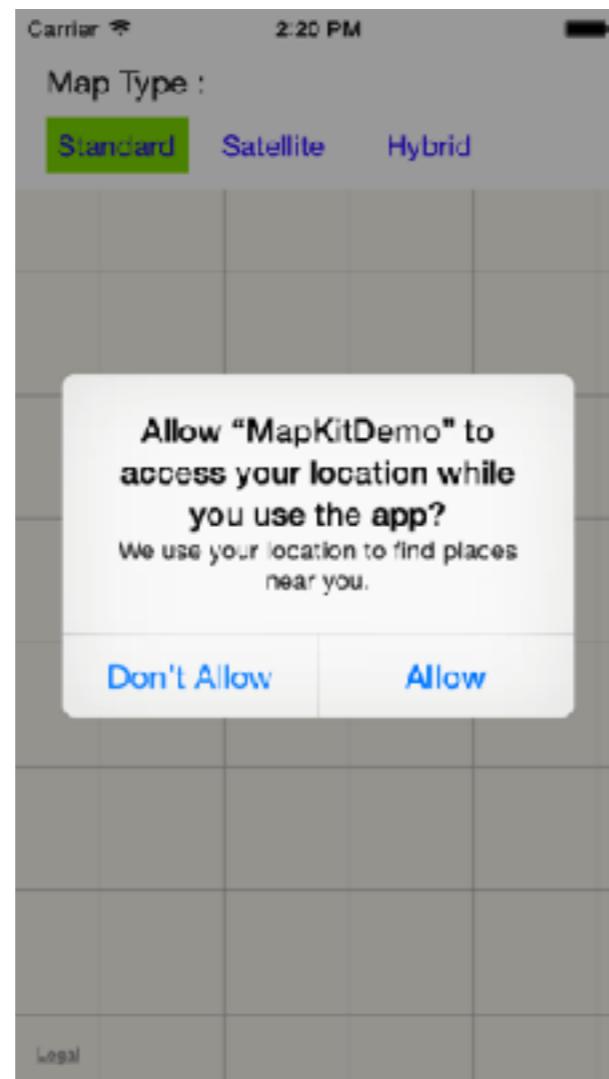
    -- need to delay for a while so that poke can appear after the map is ready
    timer.performWithDelay( 5000, mapmarker )
end
```





Project 04: Burger Map App

- Step 4: Show my location on the map
 - Location is a sensitive information, both Android and iOS require developers to ask for user permissions





Project 04: Burger Map App

- Step 4: Show my location on the map
 - Location is a sensitive information, both Android and iOS require developers to ask for user permissions
 - Declare location permission in build.settings
 - Open build.settings in your editor and add the following block

```
--  
-- Android Section  
--  
android =  
{  
    usesPermissions =  
    {  
        "android.permission.INTERNET",  
        -- Optional permission used to display current location via the GPS.  
        "android.permission.ACCESS_FINE_LOCATION",  
        -- Optional permission used to display current location via WiFi or cellular service.  
        "android.permission.ACCESS_COARSE_LOCATION",  
    },  
},
```

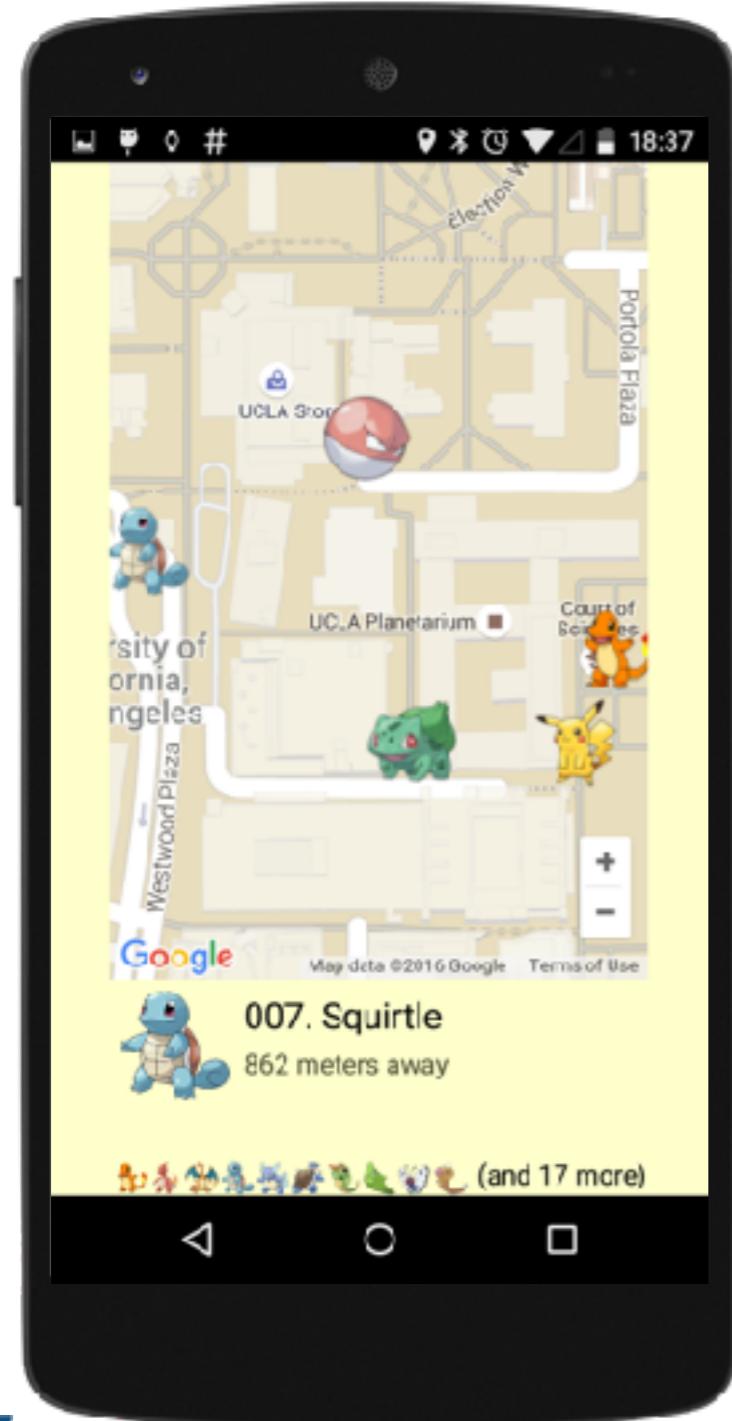


Episode 5 - The Unknown Rival

Pokemon Go Mini



Project 05: Pokemon Go Mini

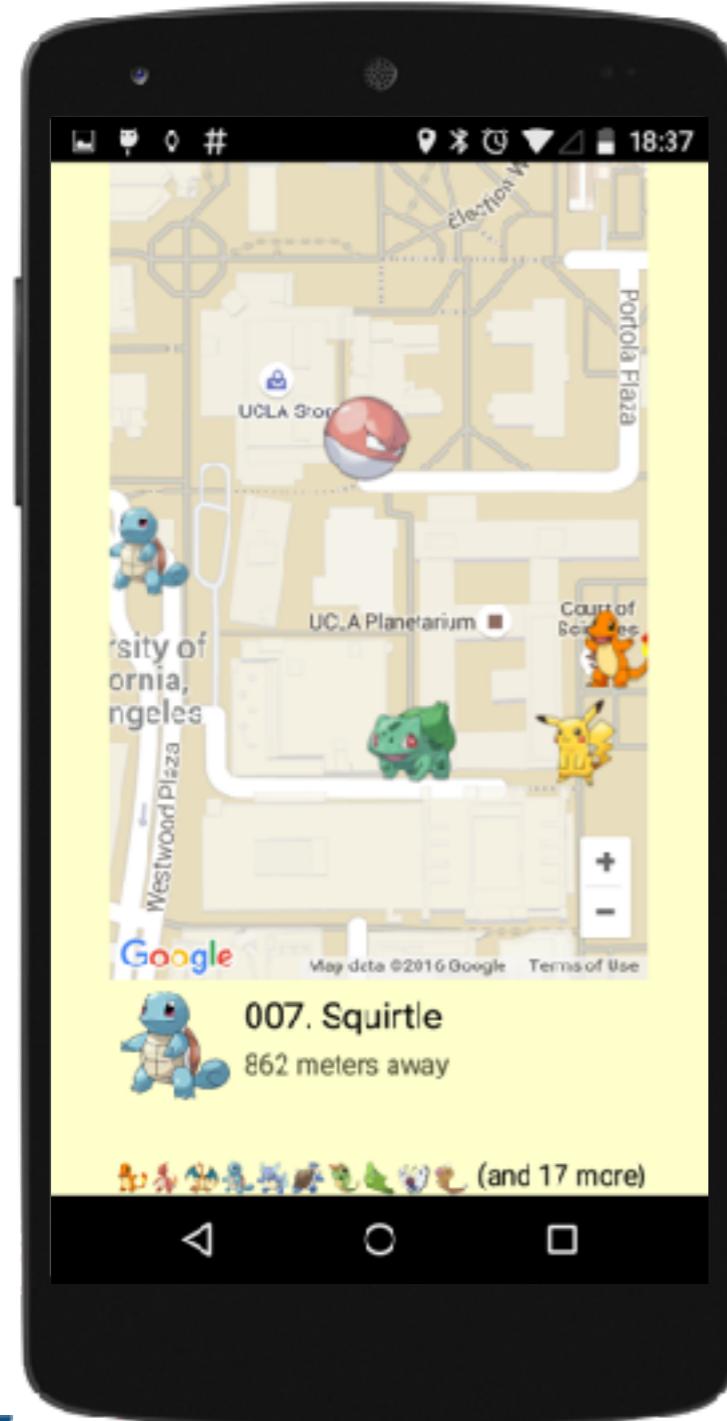


- Basic requirements
 - A map to show locations of Pokemon and yourself
 - Show distance between you and the nearest Pokemon
 - Detect Poke ball throwing action by using accelerometer





Project 05: Pokemon Go Mini



- Extra: Now it's your time to be creative!
 - Show different Pokemon at different locations every time
 - The app can automatically save the progress
 - Make a story for your app
 - Sound? Animation?
 - ...





Project 05: Pokemon Go Mini

- Step ?: Put images on a map

```
local opt1 =  
{  
    imageFile = "image.png"  
}  
myMap:addMarker(34.063327, -118.445130, opt1 )
```

Homework submission

- Create a folder called FirstName_LastName_Module3
 - Inside, there should be 5 folders: project1, project2, ..., project5
- Compress the outside folder and submit to the dropbox link (to be appeared)
- (Optional) A report to show your “selling points” of your projects, itemize them, each project no more than 5 items



Research topics

- What is rooting an Android phone? Why do you need it? How do you do it?
- Augmented reality and Google Cardboard
- Google glass - why and why not?
- Smart watch and fitness trackers
- Different radios on a smartphone: Wifi, LTE/4G, Bluetooth, NFC, etc.
- Integrated Development Environment (IDE): Eclipse ADT vs Android Studio
- Extending battery life and Android WakeLock





*Send any question to
bojhang@cs.ucla.edu*