



Introduction to Mobile Application - Android Programming with Corona

Luis Garcia
Networked & Embedded Systems Lab (NESL)
ECE Department, UCLA
garcialuis@ucla.edu
July 17-18, 2019



UCLA

Module Acknowledgement

- Bo-Jhang Ho, *NESL Hall of Famer*



I think you're ready !!

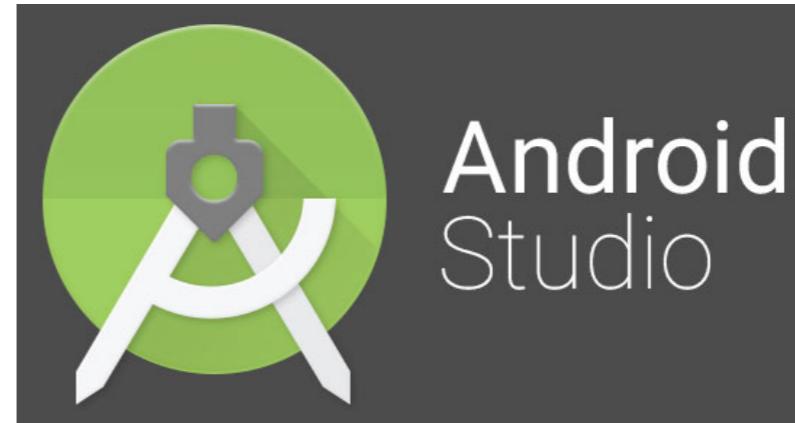


First things first



Corona SDK

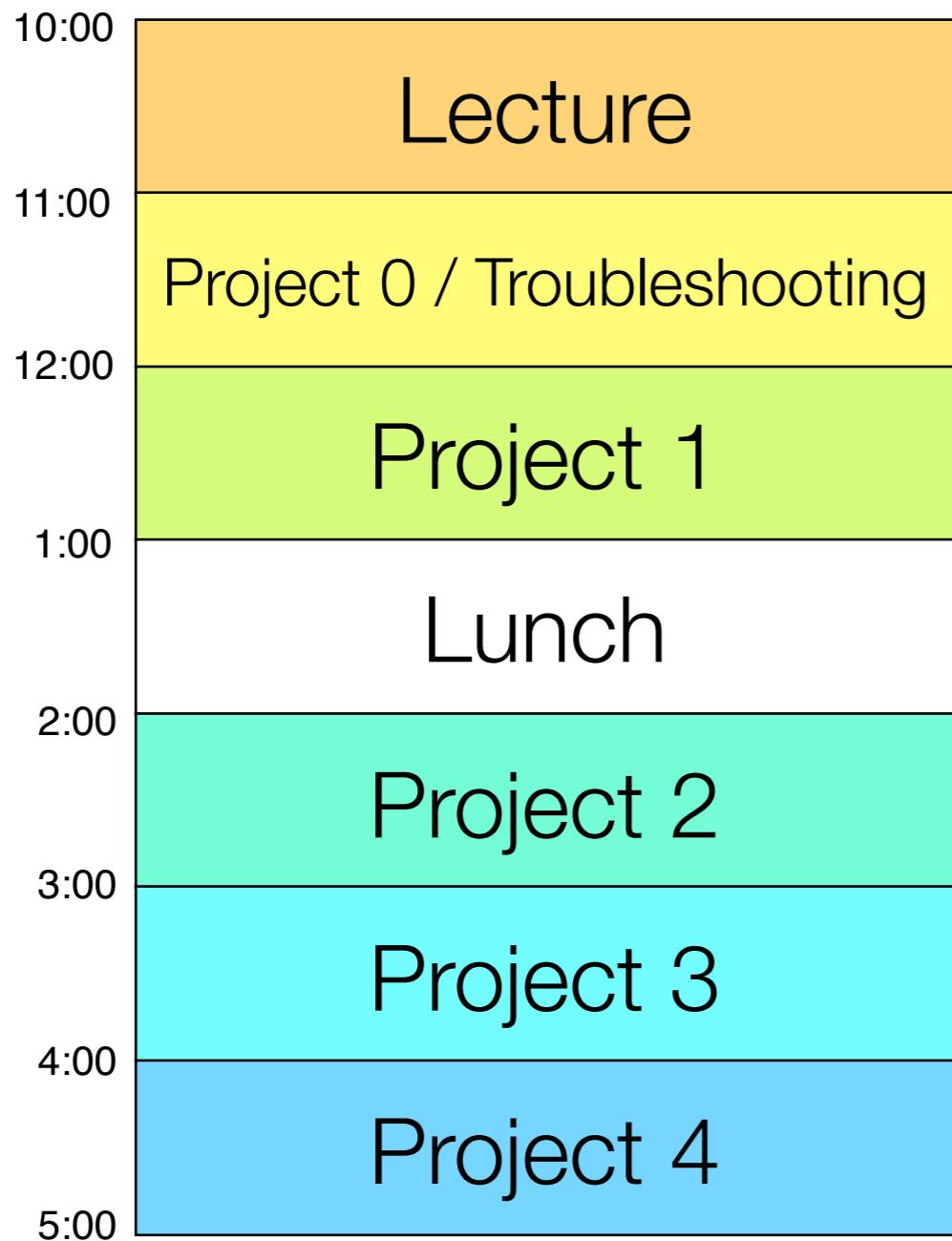
1. Download this



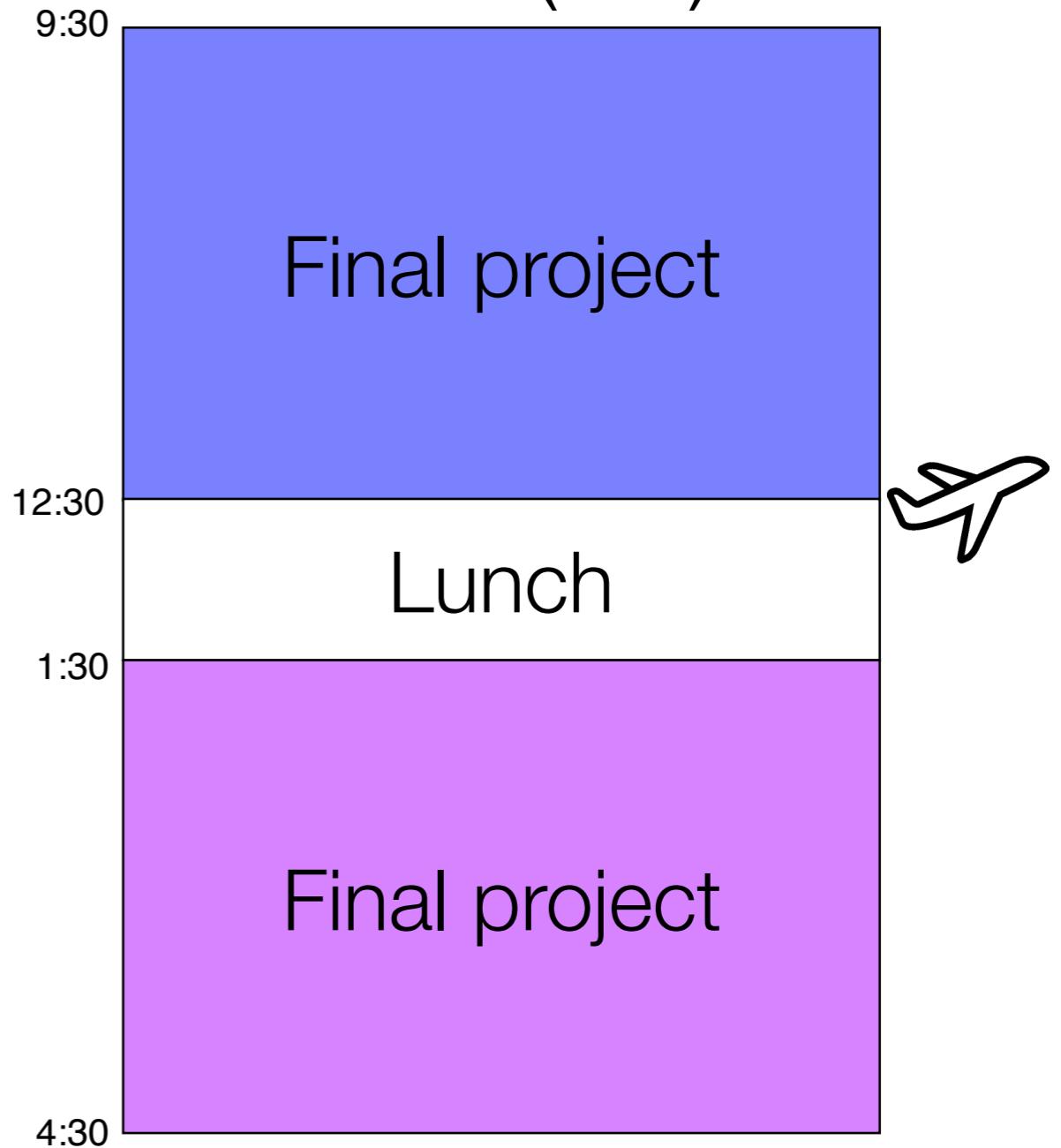
2. Download this

Agenda

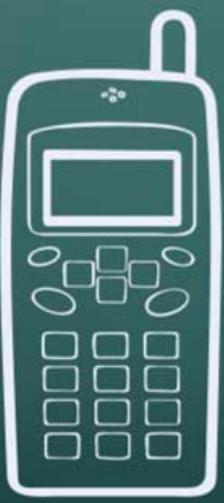
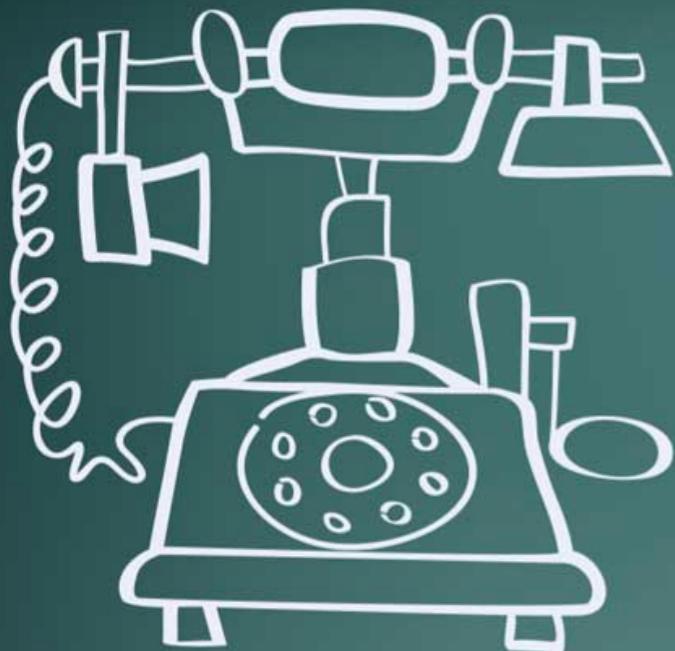
Jul. 18 (Mon)



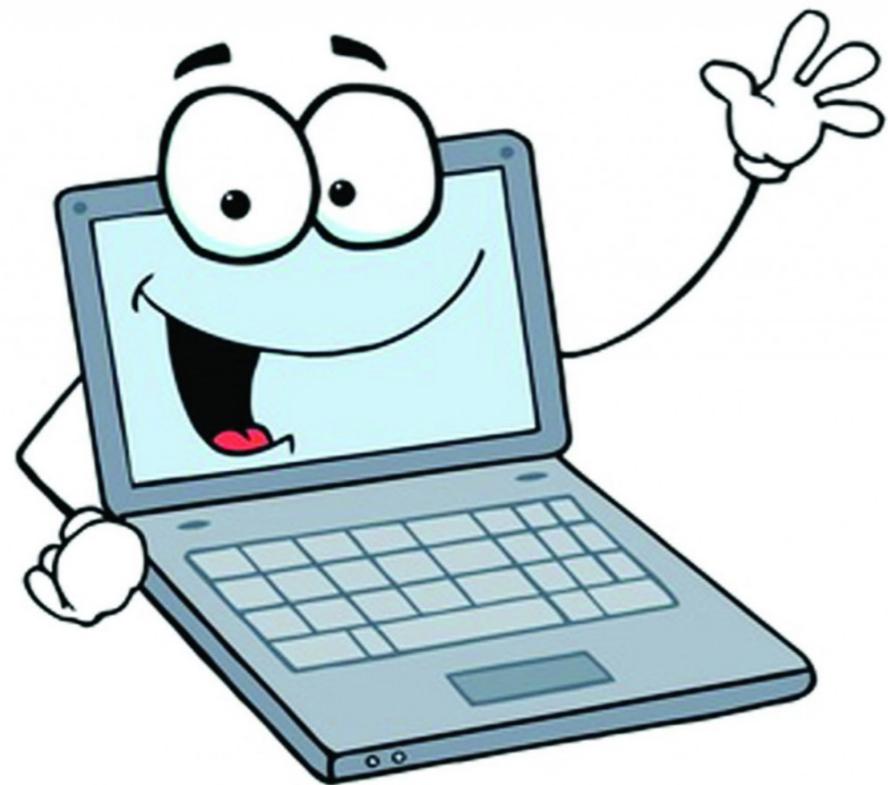
Jul. 19 (Tue)



Is a smartphone a phone?



Difference between a computer and a mobile phone nowadays??



KidsToday.in

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?

Work at one place

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
 - touch, speech, haptic, blinking, ...
- Many built-in sensors
 - Measure motion, orientation, location, temperature, ...
- Radios for communication
 - Bluetooth, WiFi access points, NFC, ...
- Cameras + powerful CPU / GPU to enable augmented reality
 - Deep learning!



Mobile applications

- Why are smartphones so popular? What do you love them for?



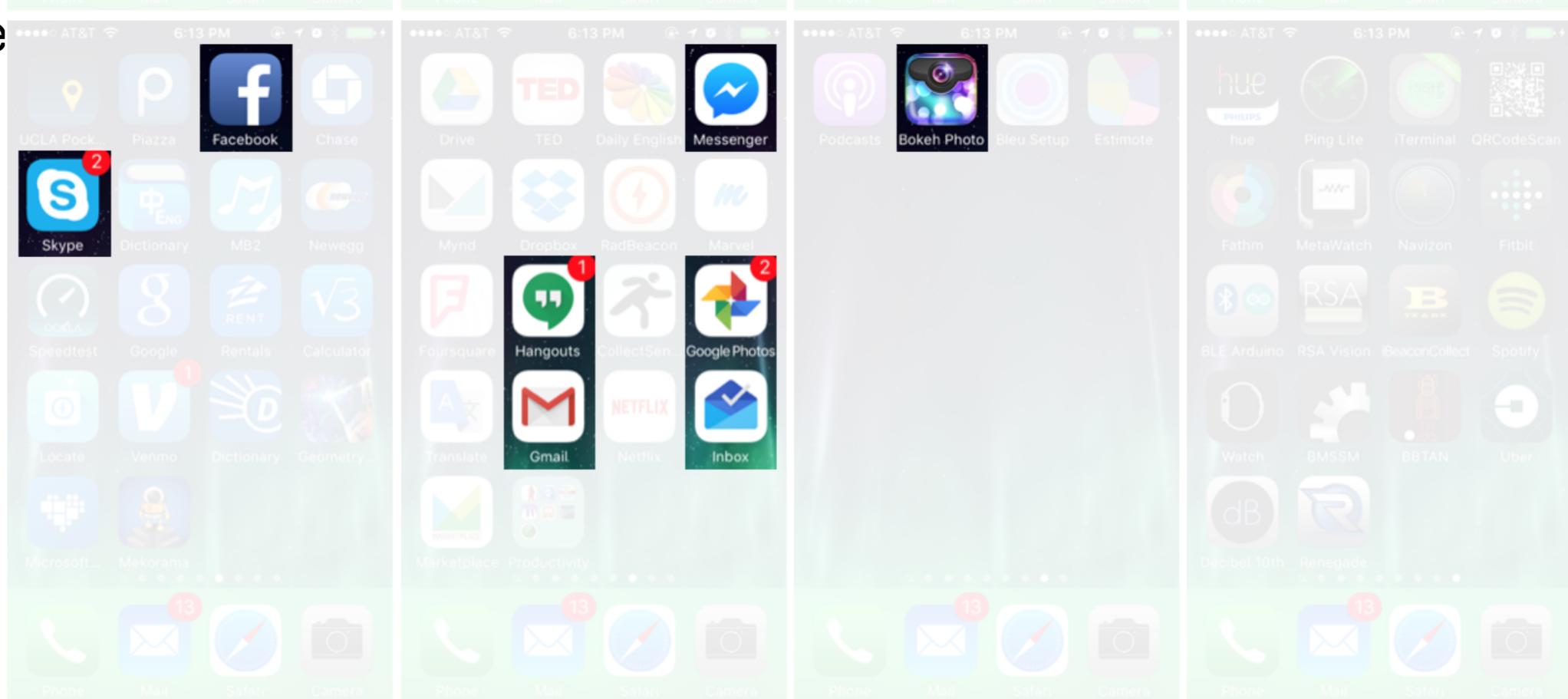
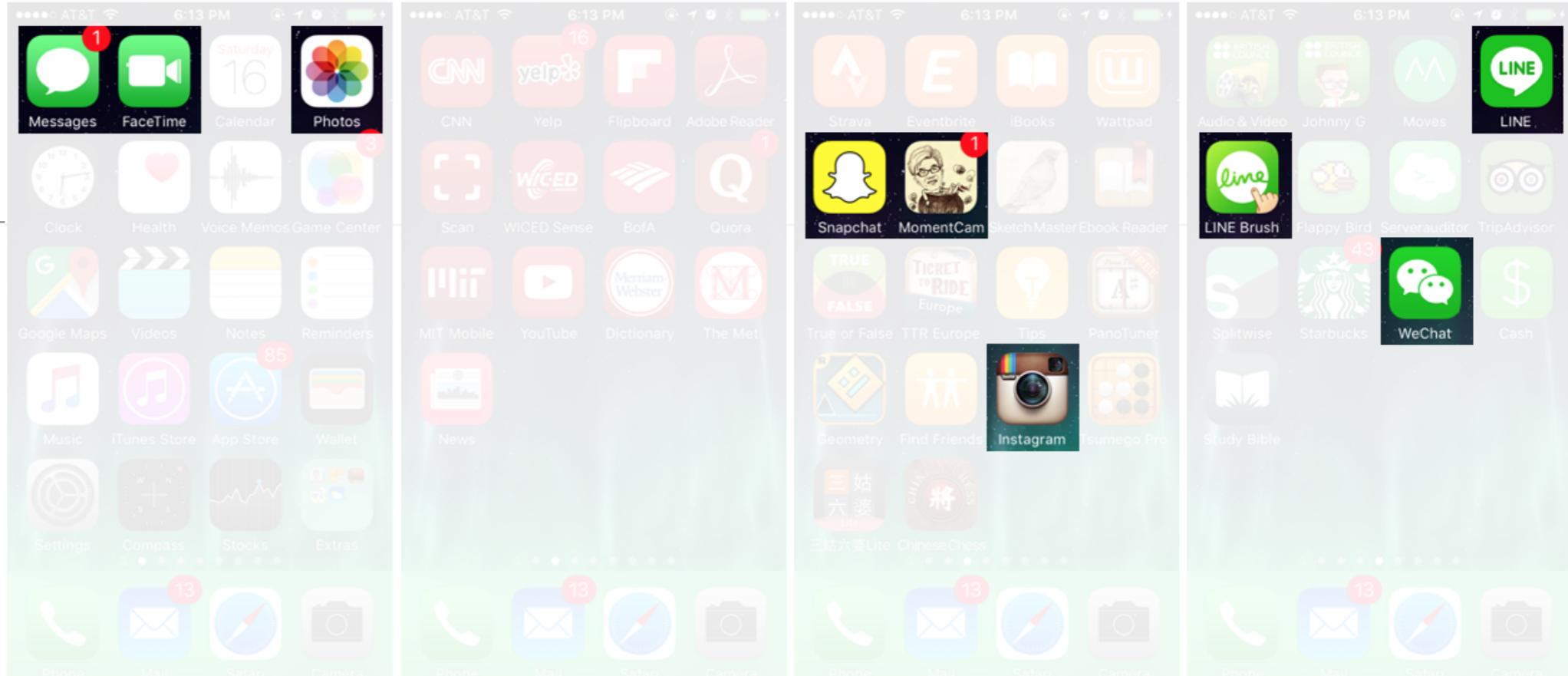
Real case study



Real case study

- **Social**

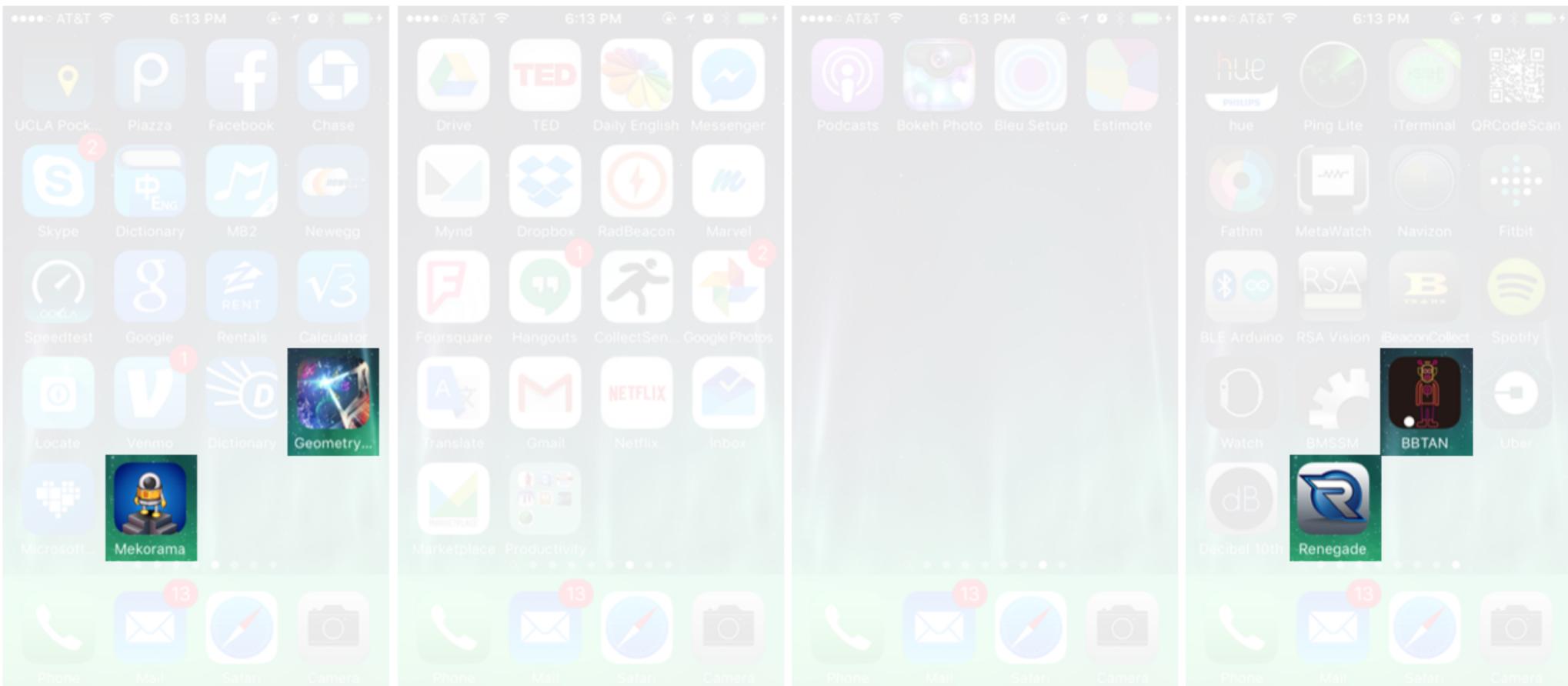
- Instant message
- Networking
- Photo sharing



Real case study

- Games

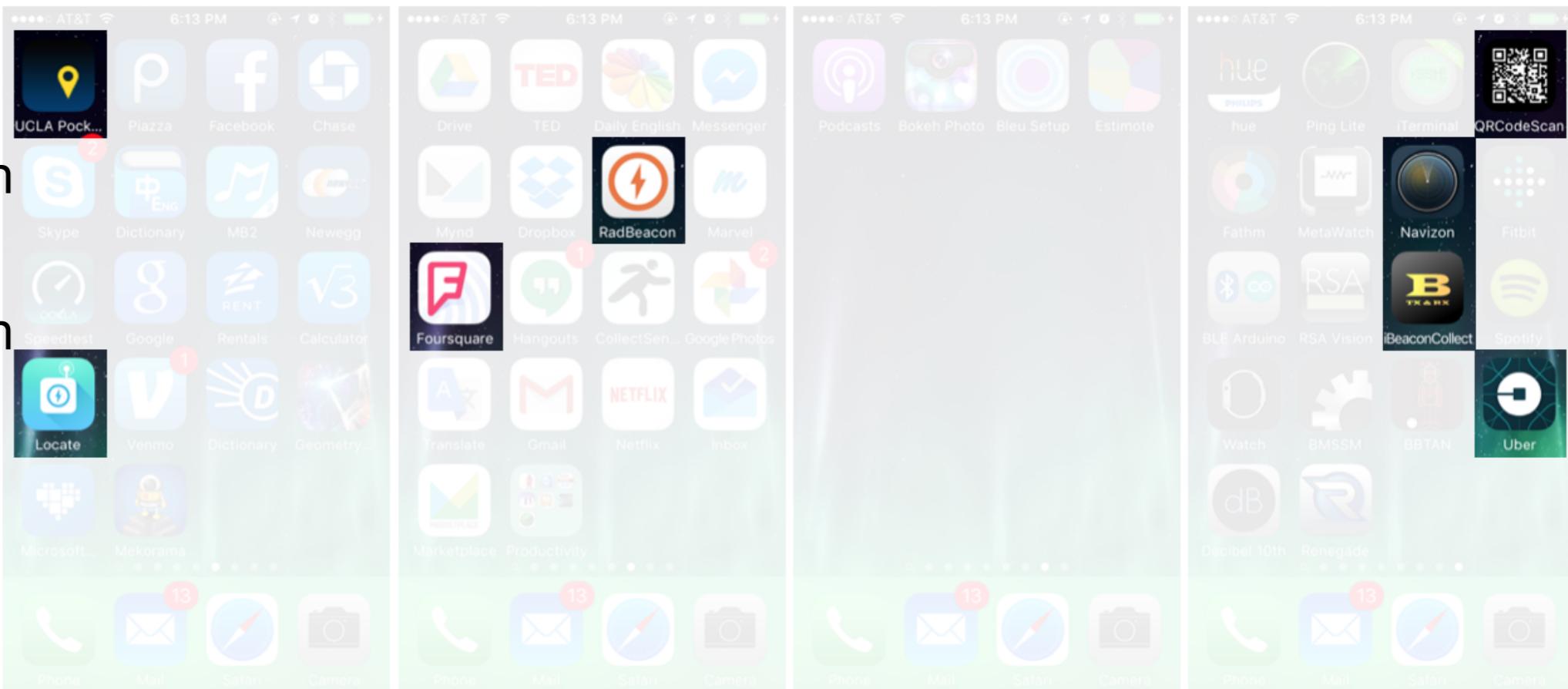
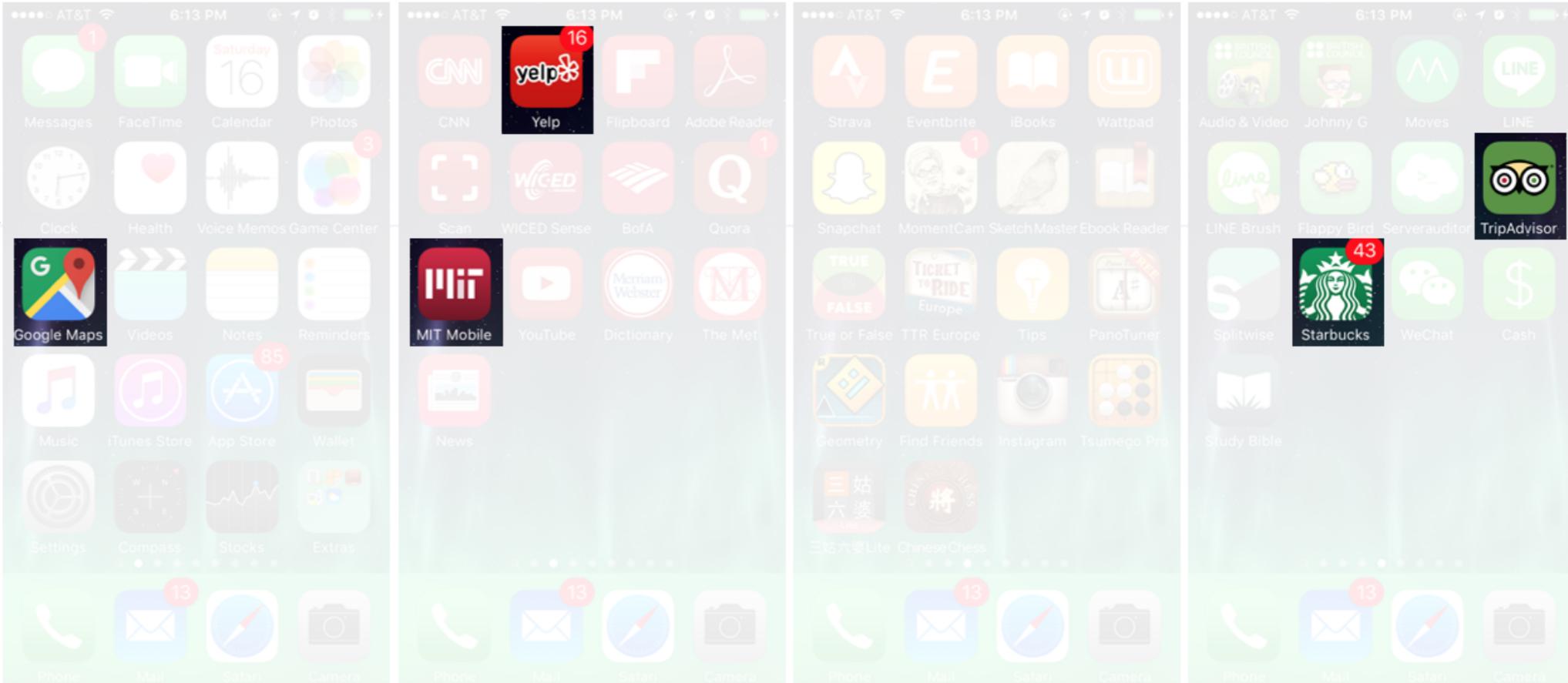
- Do I need to say more?



Real case study

- **Location service**

- Maps
- Restaurant recommendation
- Travel recommendation
- Indoor navigation



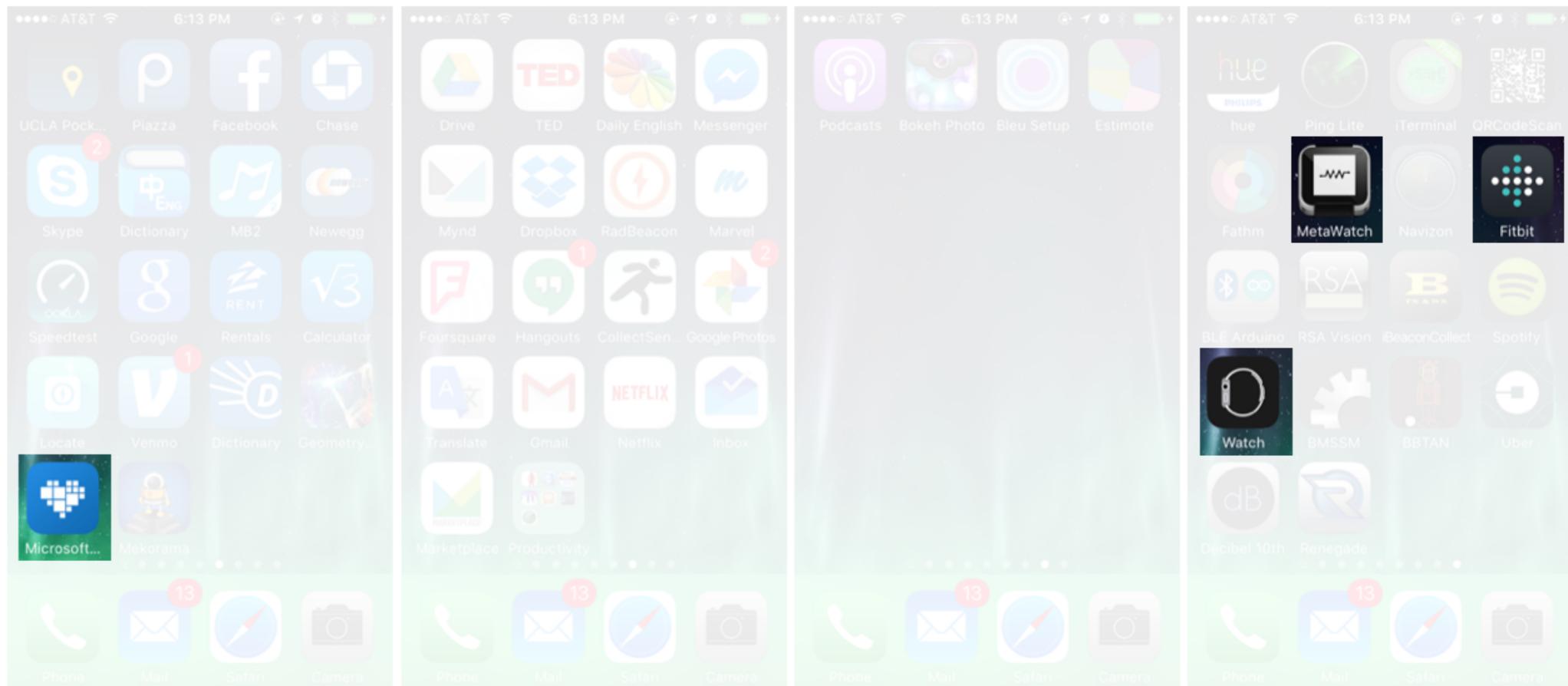
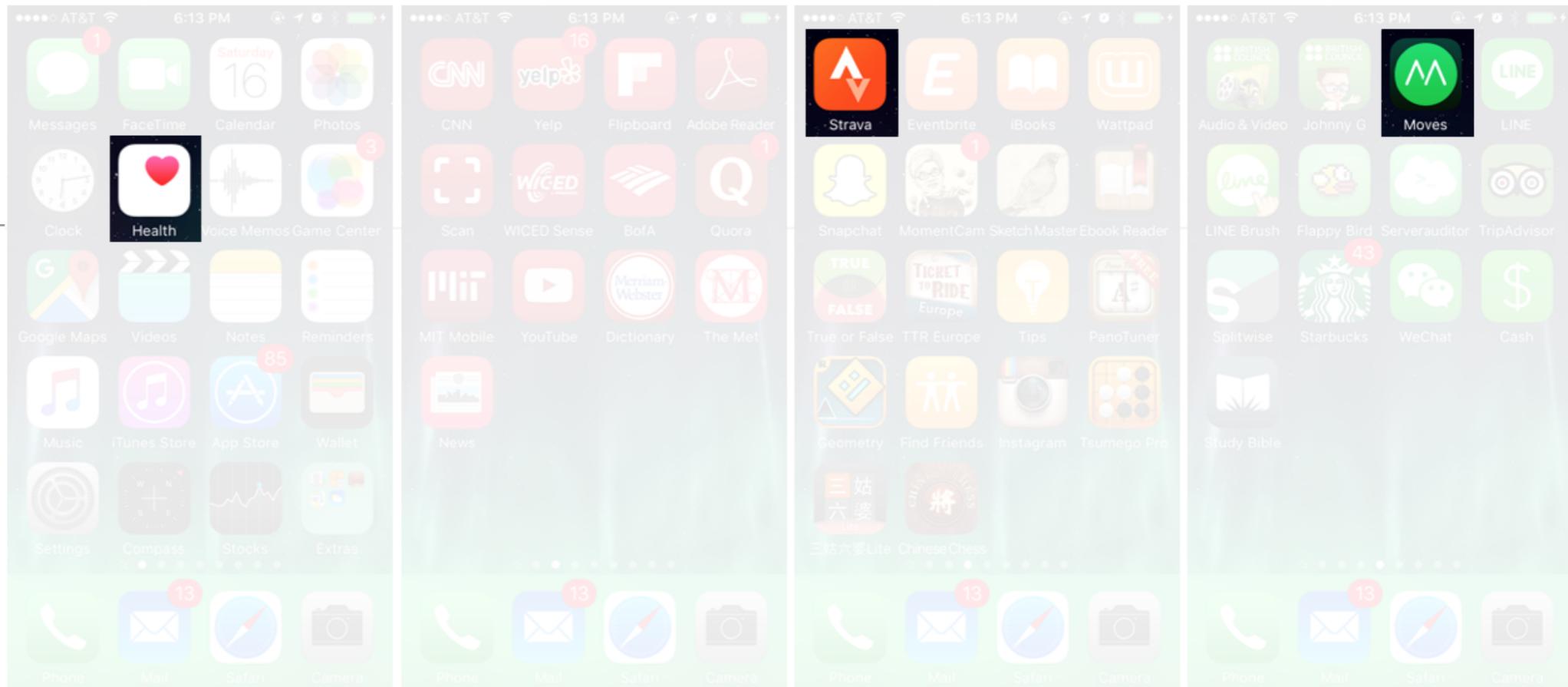
Real case study

- Learning
 - News
 - Readings
 - Dictionaries
 - Piazza!



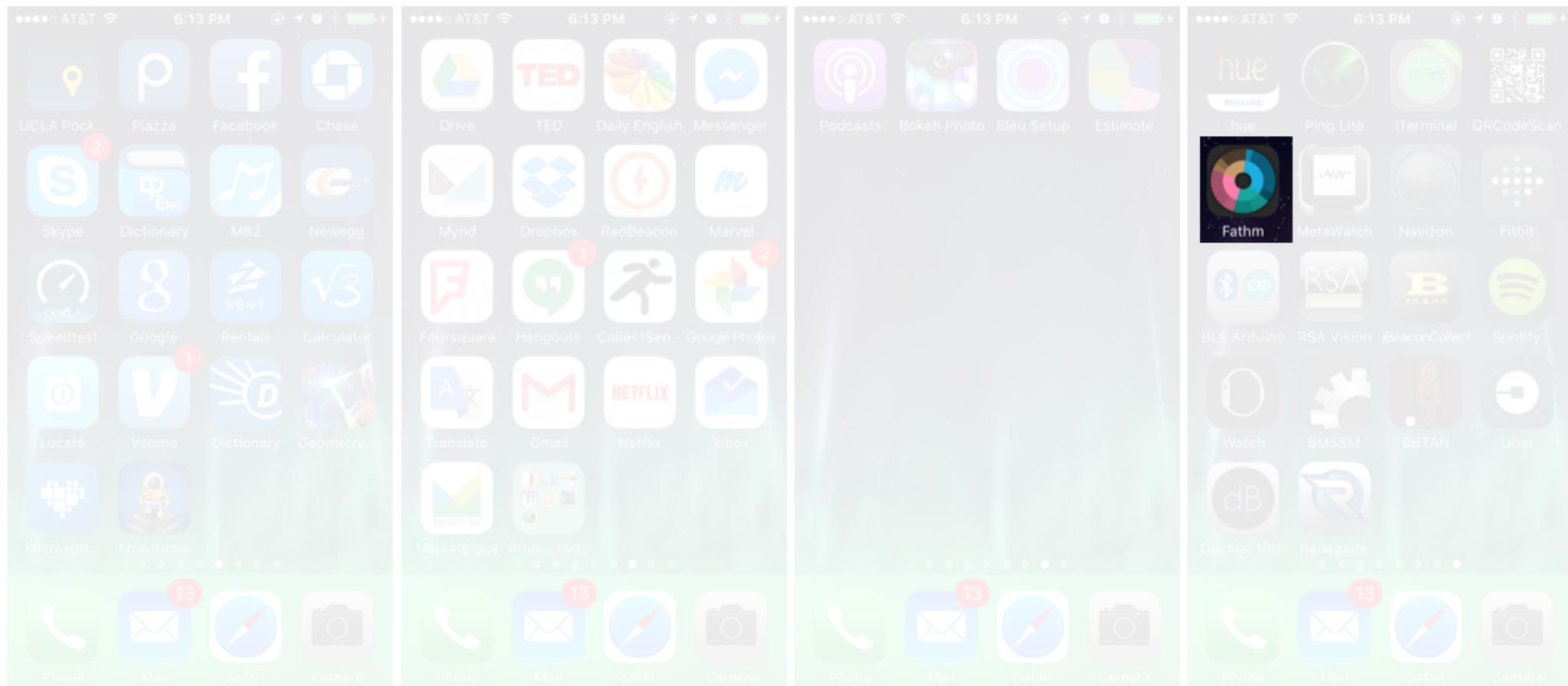
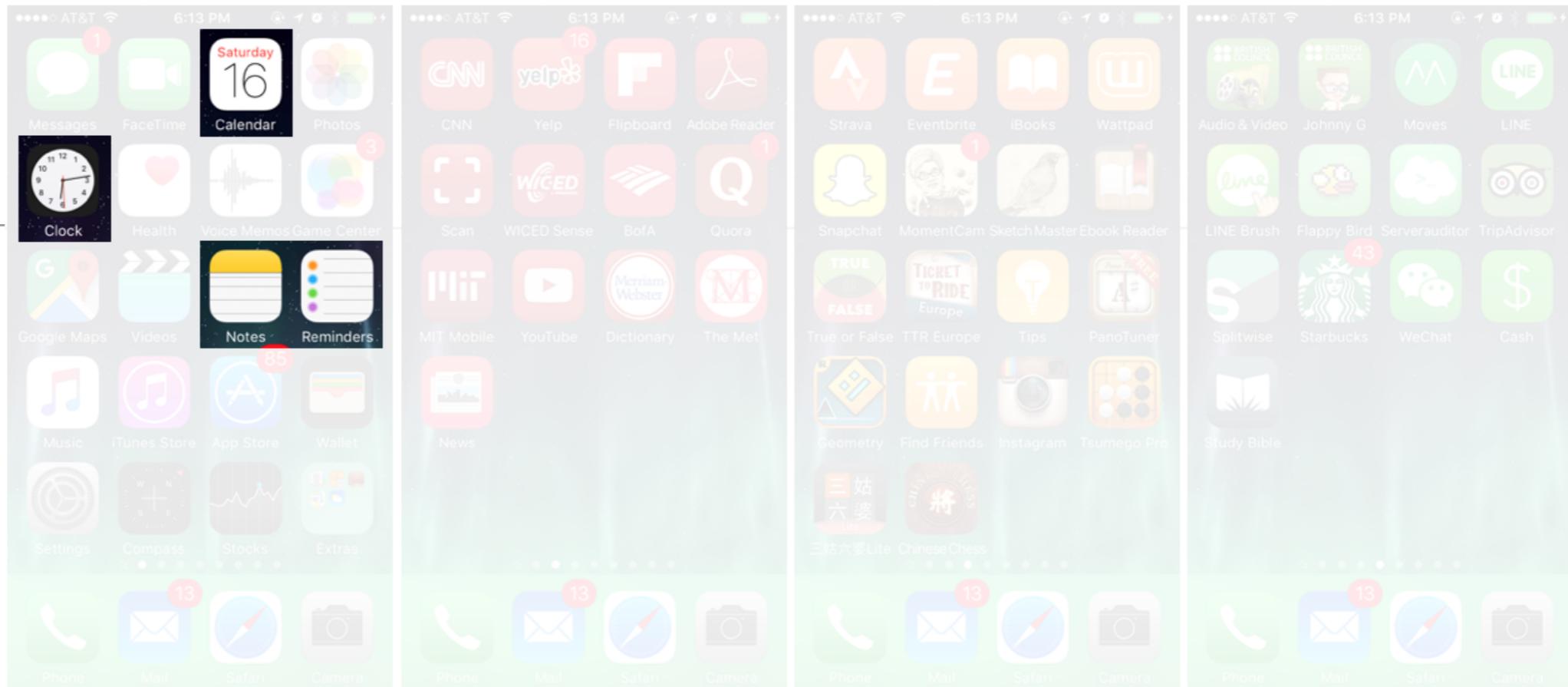
Real case study

- Health
 - Workout tracking



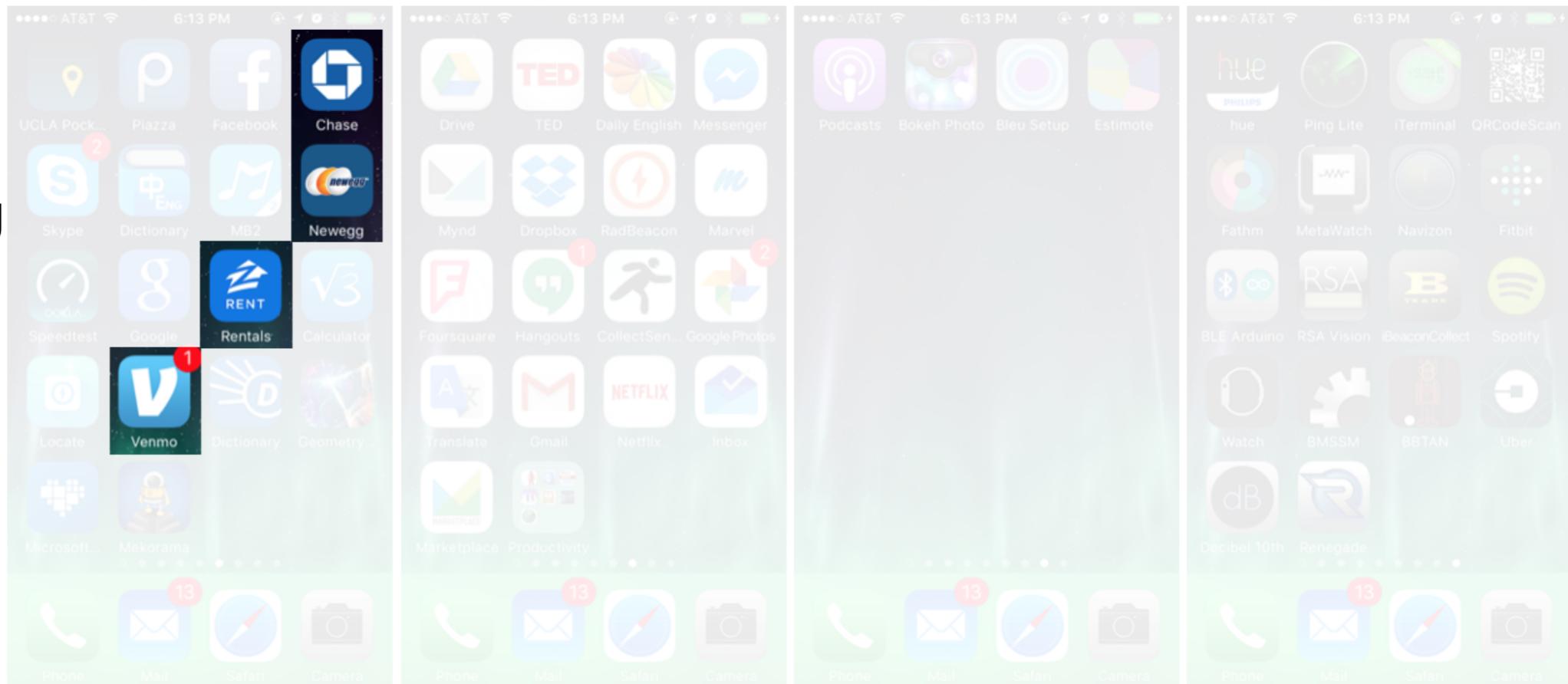
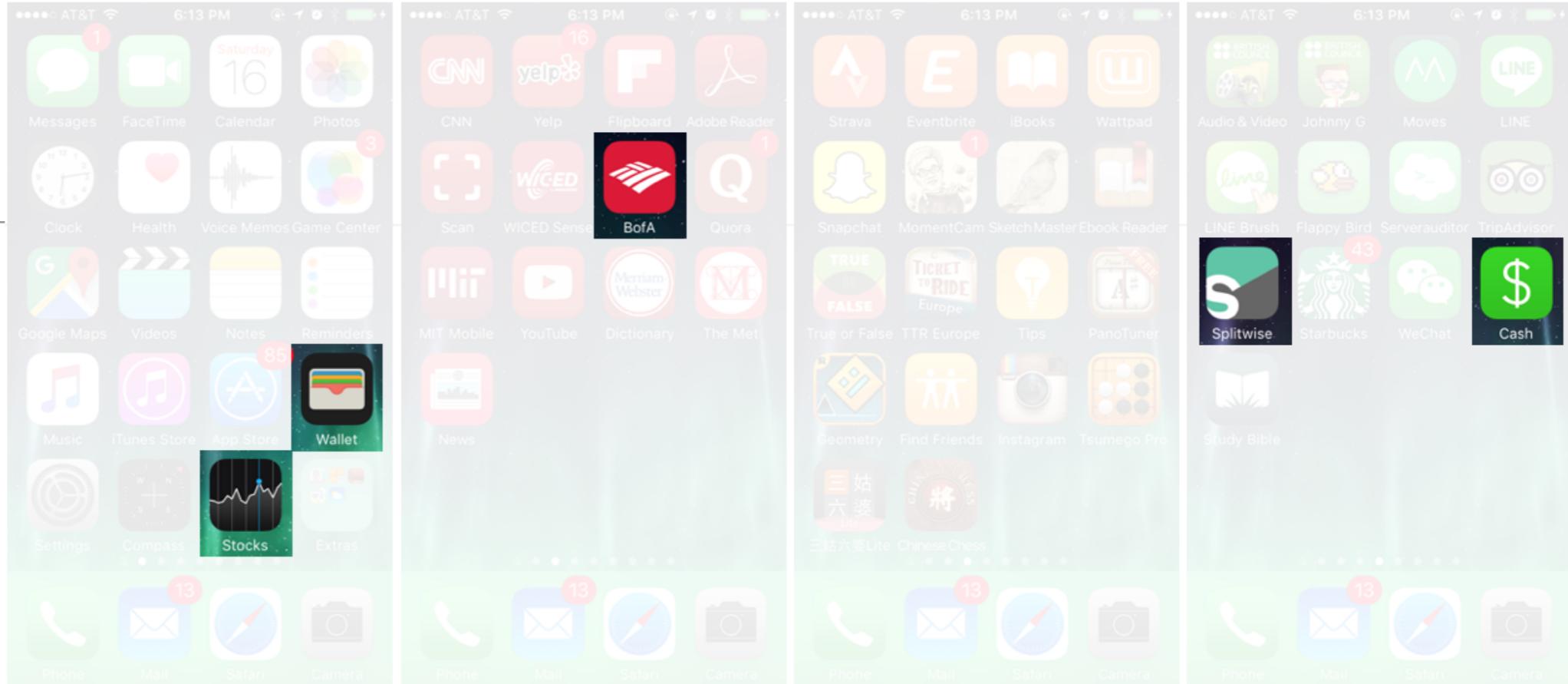
Real case study

- **Scheduling**
 - Calendar
 - Reminder



Real case study

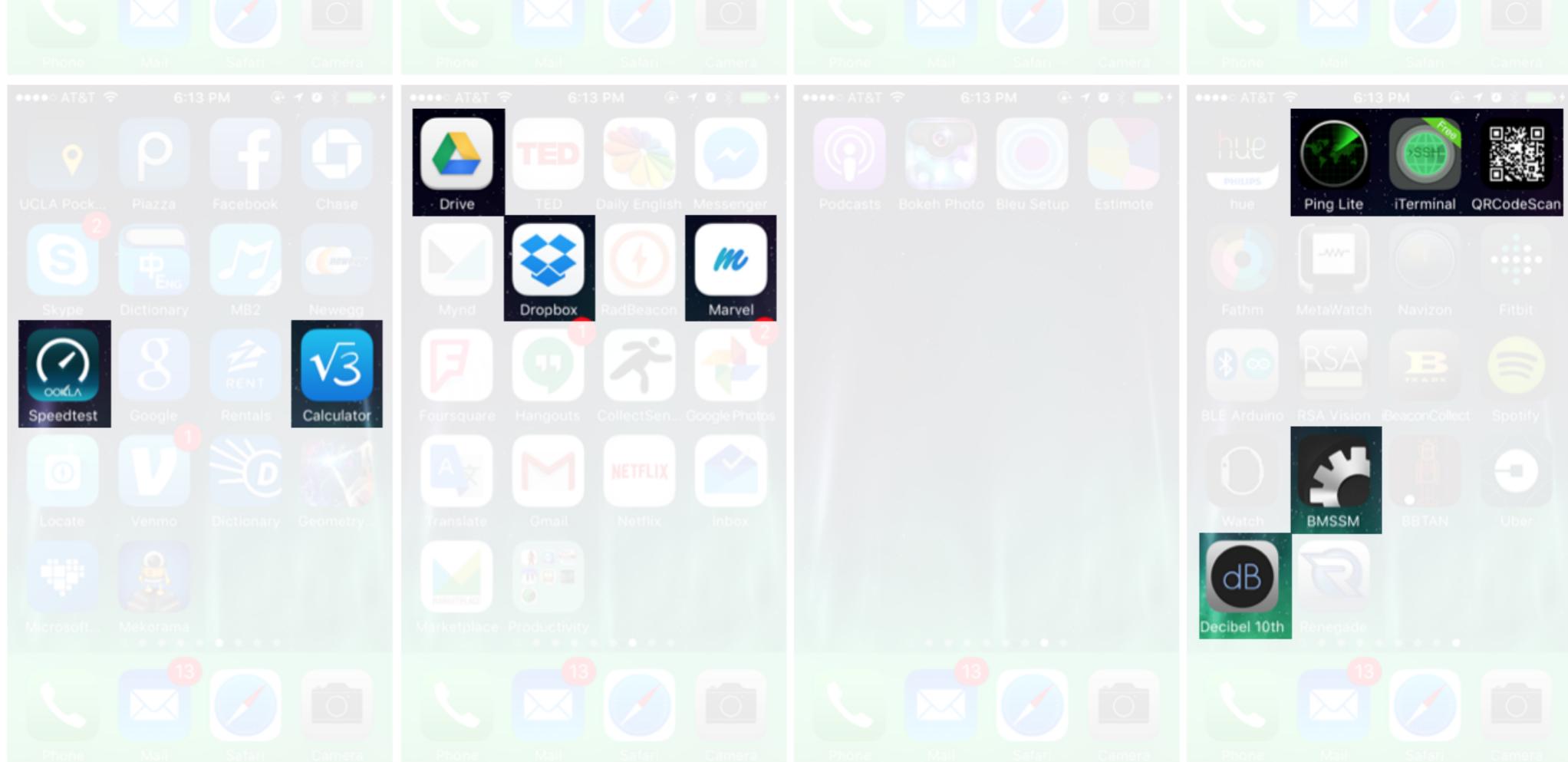
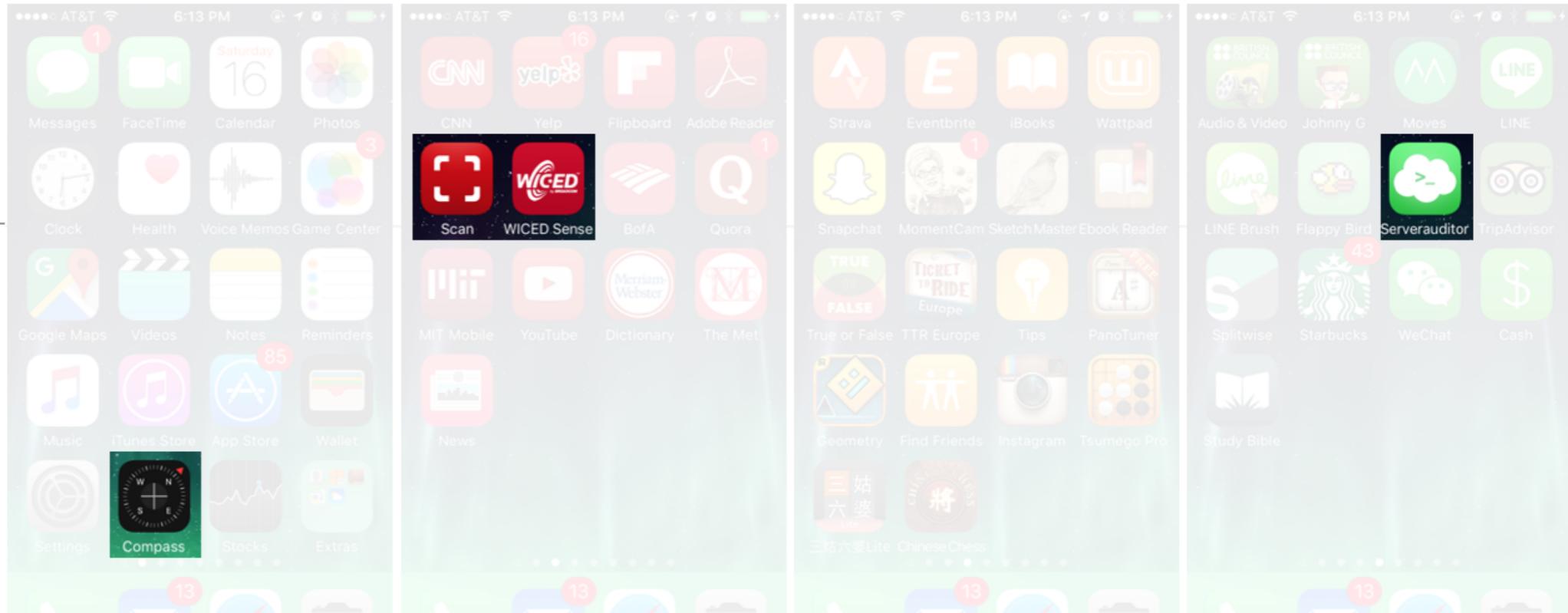
- Financial
 - Stock
 - Banking
 - Online shopping



Real case study

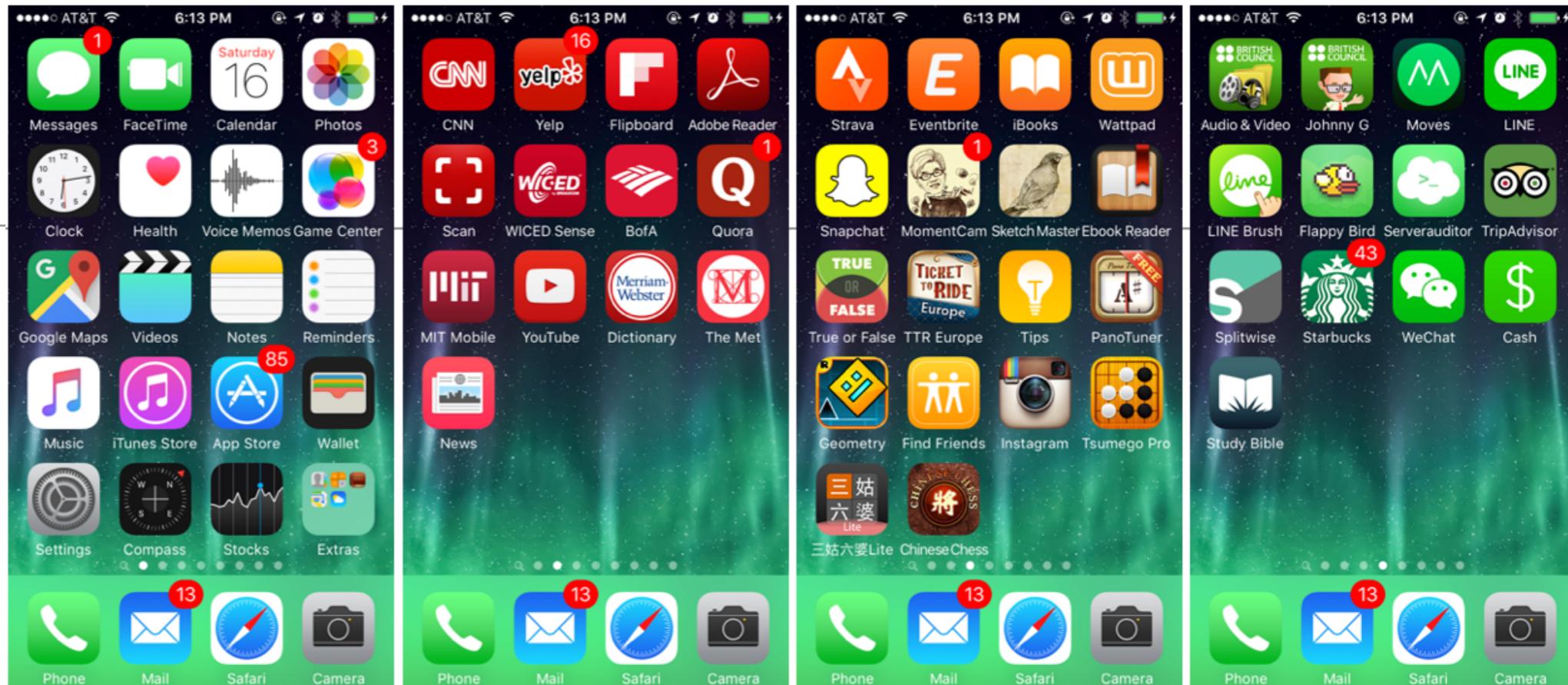
- **Productive**

- Tools
- Online storage



Real case study

- Whoa! That's why you need your phone every day!



Popular mobile app development model



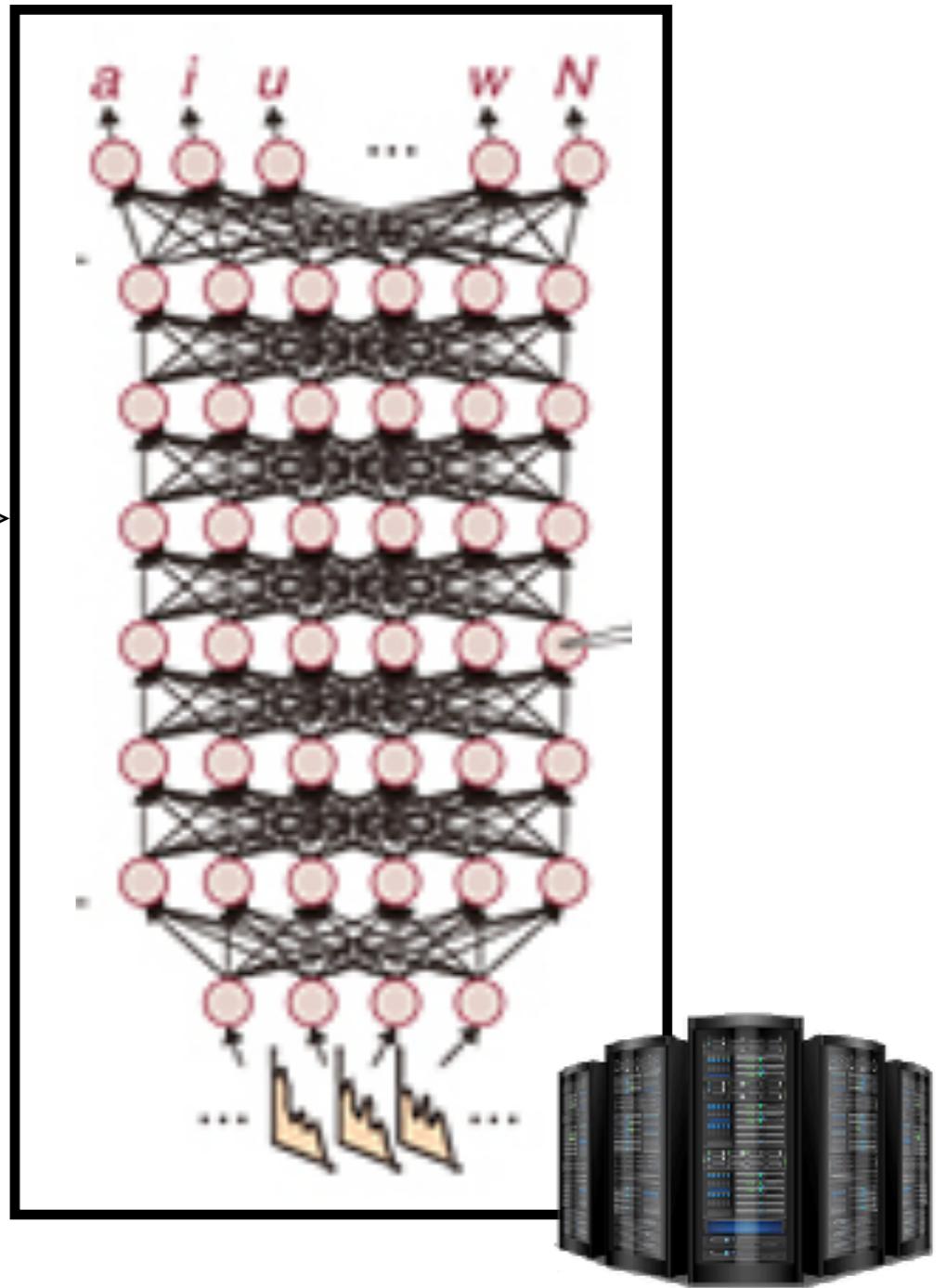
Popular mobile app development model



Popular mobile app development model



Audio data
→
←
Text-based command



Mobile development ecosystem

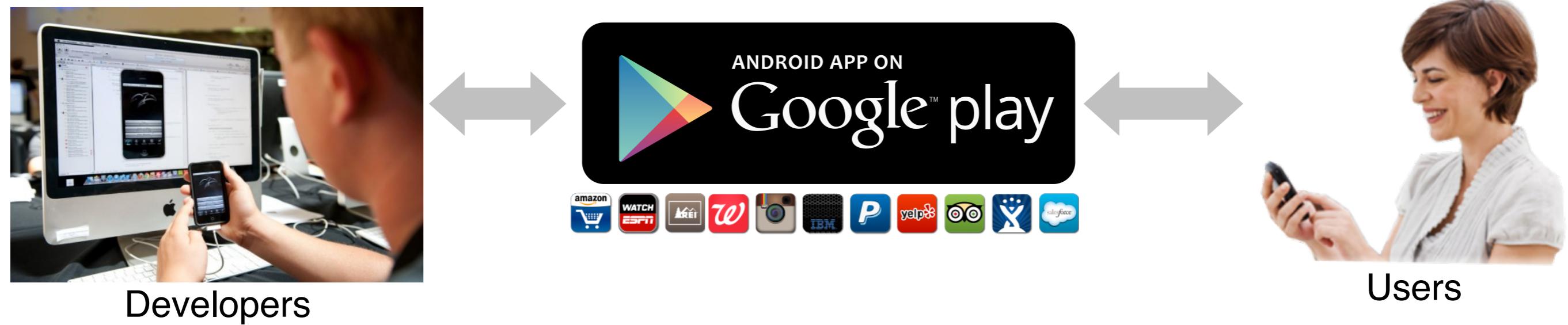


Mobile development ecosystem

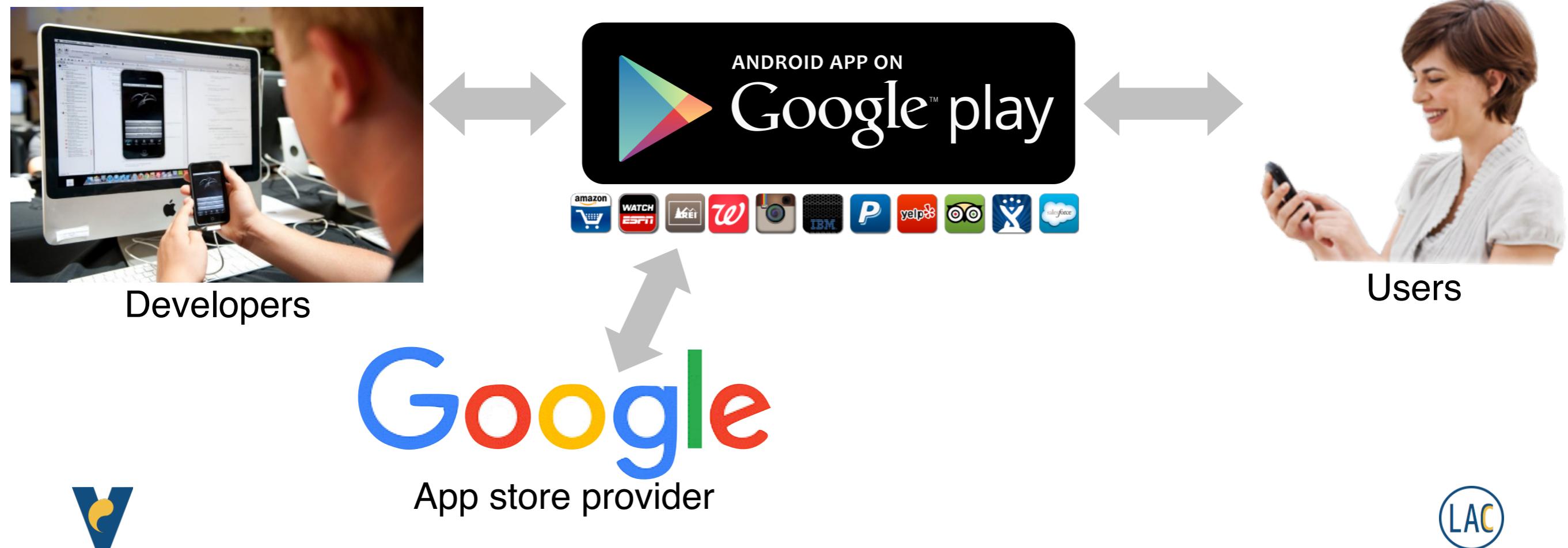


Users

Mobile development ecosystem



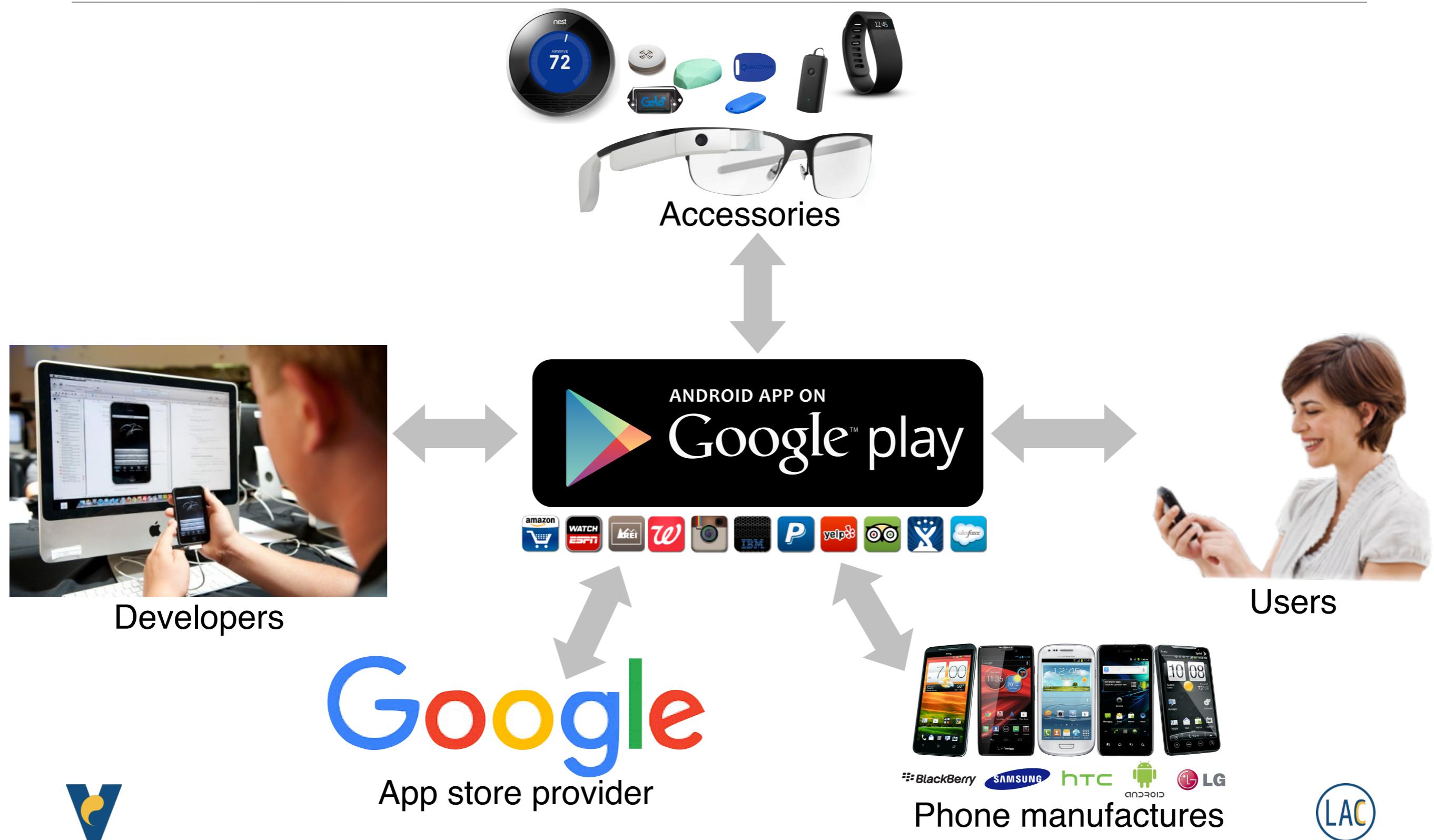
Mobile development ecosystem



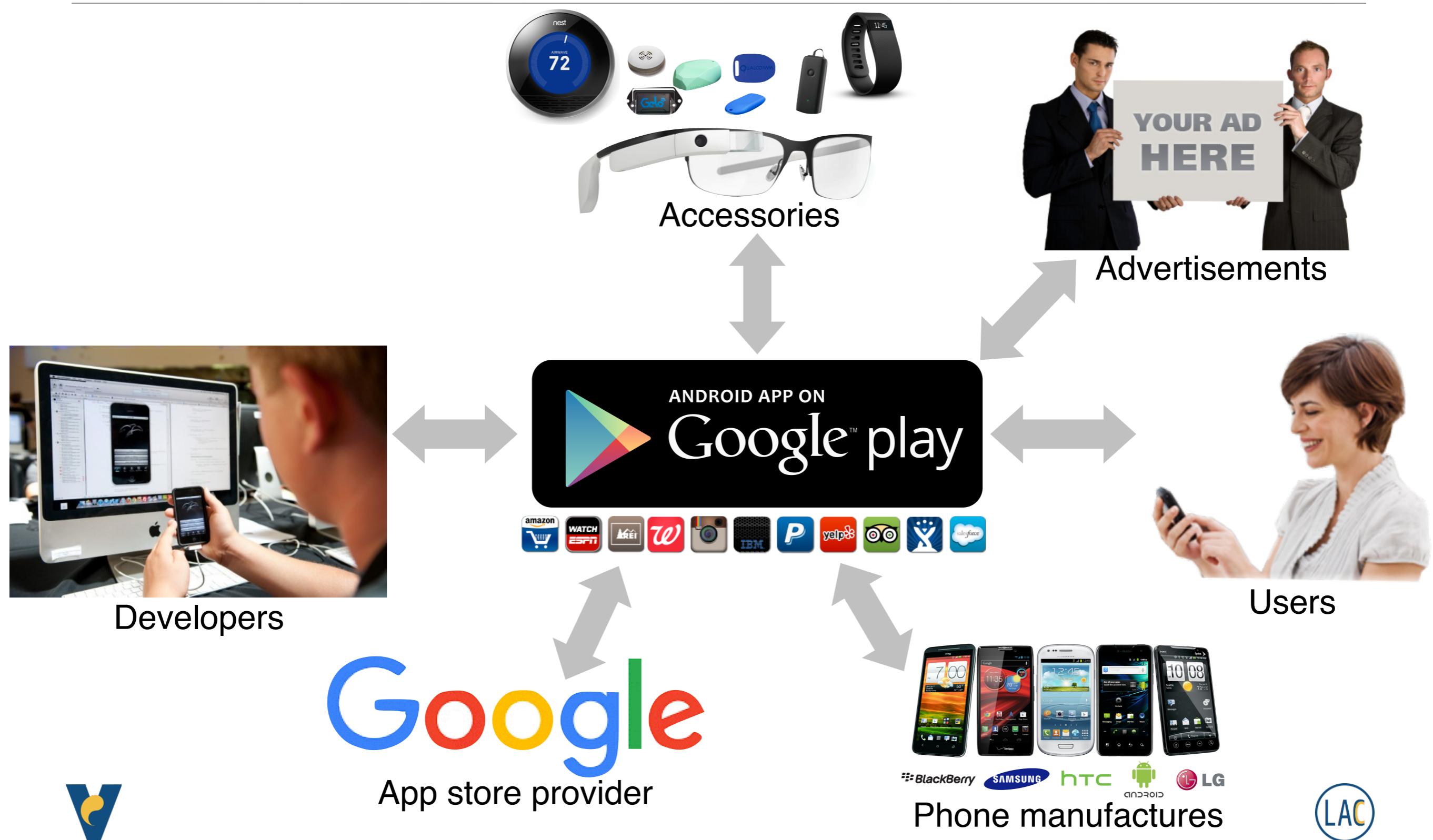
Mobile development ecosystem



Mobile development ecosystem



Mobile development ecosystem

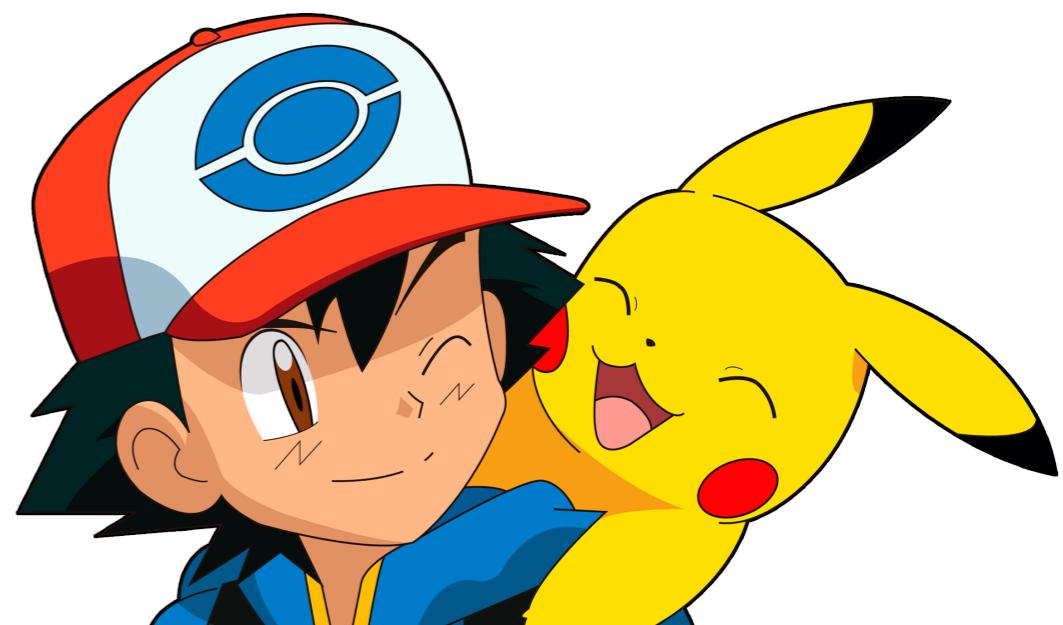


Mobile development ecosystem



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

Android app development using Corona SDK



Popular mobile development platforms



Popular mobile development platforms



Java



Objective-C
Swift



Popular mobile development platforms

The screenshot shows the Android Studio IDE interface. The main area displays Java code for the `MainActivity`. The code handles button clicks to register a device or start a stream, and it inflates a menu bar. The project structure on the left shows files like `activity_main.xml`, `MainActivity.java`, and `StreamActivity.java`. The bottom navigation bar includes tabs for Home, Recent, and Help.

```
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);
                startActivity(intent);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```



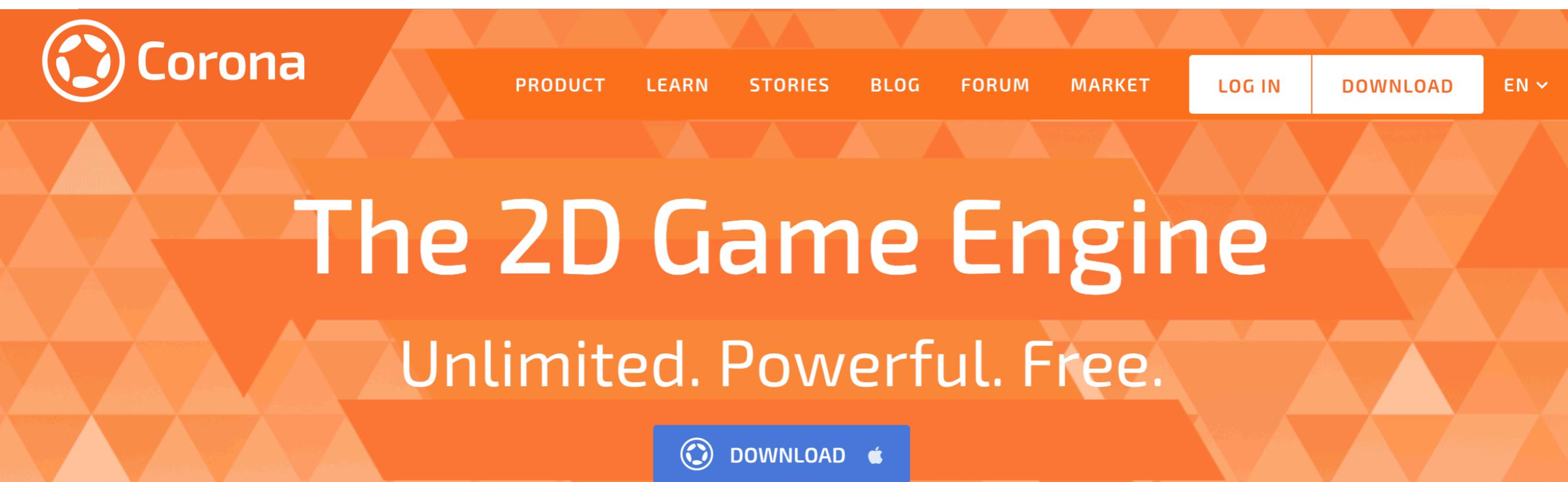
Popular mobile development platforms

The screenshot shows the Xcode interface with three main panes:

- Left Pane:** Displays Swift code for a game scene setup. The code includes functions for Blimp Control, Scene Configuration, and Scene Initialization, along with logic for handling physics contacts.
- Middle Pane:** Shows the game's preview window titled "Balloons". It depicts a colorful scene with a blimp, balloons, and a wind field effect.
- Bottom Right Pane:** Displays a graph of a sine wave with the equation $y = 80 \cdot \sin(x)$.

A lot of code!!

Why do we choose Corona SDK?



The screenshot shows the official Corona SDK website. The header features a white navigation bar with the Corona logo (a stylized ball icon) and the word "Corona" in white. To the right are links for "PRODUCT", "LEARN", "STORIES", "BLOG", "FORUM", and "MARKET". On the far right are "LOG IN" and "DOWNLOAD" buttons, along with a language selector "EN ▾". The main content area has a large orange background with a geometric triangle pattern. It displays the text "The 2D Game Engine" in large white font, followed by "Unlimited. Powerful. Free." in a slightly smaller white font. At the bottom of this section is a blue "DOWNLOAD" button with the Corona logo and an Apple logo.

The 2D Game Engine

Unlimited. Powerful. Free.

 DOWNLOAD 

- Plus, you can compile on both iOS and Android platforms!
- No need to deal with the complex library and build systems
- Provides 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 Run, along with tabs for the current project (streamplayer), file (activity_main.xml), and class (MainActivity.java). The left sidebar displays the Project structure, showing the main project folder (streamplayer) containing subfolders like .idea, assets, bin, gen, libs, res (with drawable-hdpi, drawable-ldpi, drawable-mdpi, drawable-xhdpi, drawable-xxhdpi, and layout subfolders), menu, values, and xml; and a src folder with a co.infinum.streamplayer package containing MainActivity, StreamActivity, StreamApp, and UserSettingsActivity classes, along with build-related files (.classpath, .project, AndroidManifest.xml, build.gradle, combat.yaml, ic_launcher-web.png, proguard-project.txt, project.properties, streamplayer.iml). The right side of the screen is the code editor, currently displaying the MainActivity.java file. The code implements an Activity that handles button clicks to register a device or start a stream, and also handles options menu items for settings and user settings.

```
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.streamplayer.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);
                startActivity(intent);
                break;
            case R.id.menu_exit:
                finish();
                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 structure, including the main source code folder 'streamoplayer' containing 'MainActivity.java', 'StreamActivity.java', and 'UserSettingsActivity.java'. It also includes resource folders like 'res' (with drawable and layout subfolders) and 'src' (with 'co.infinum.streamoplayer' package). The right side of the screen features the Java code editor for 'MainActivity.java'. The code defines the MainActivity class, which extends Activity. It includes methods for handling button clicks and menu options, such as 'onCreateOptionsMenu' and 'onOptionsItemSelected'. A red arrow points from the word 'Run' at the top to the green 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);
        });

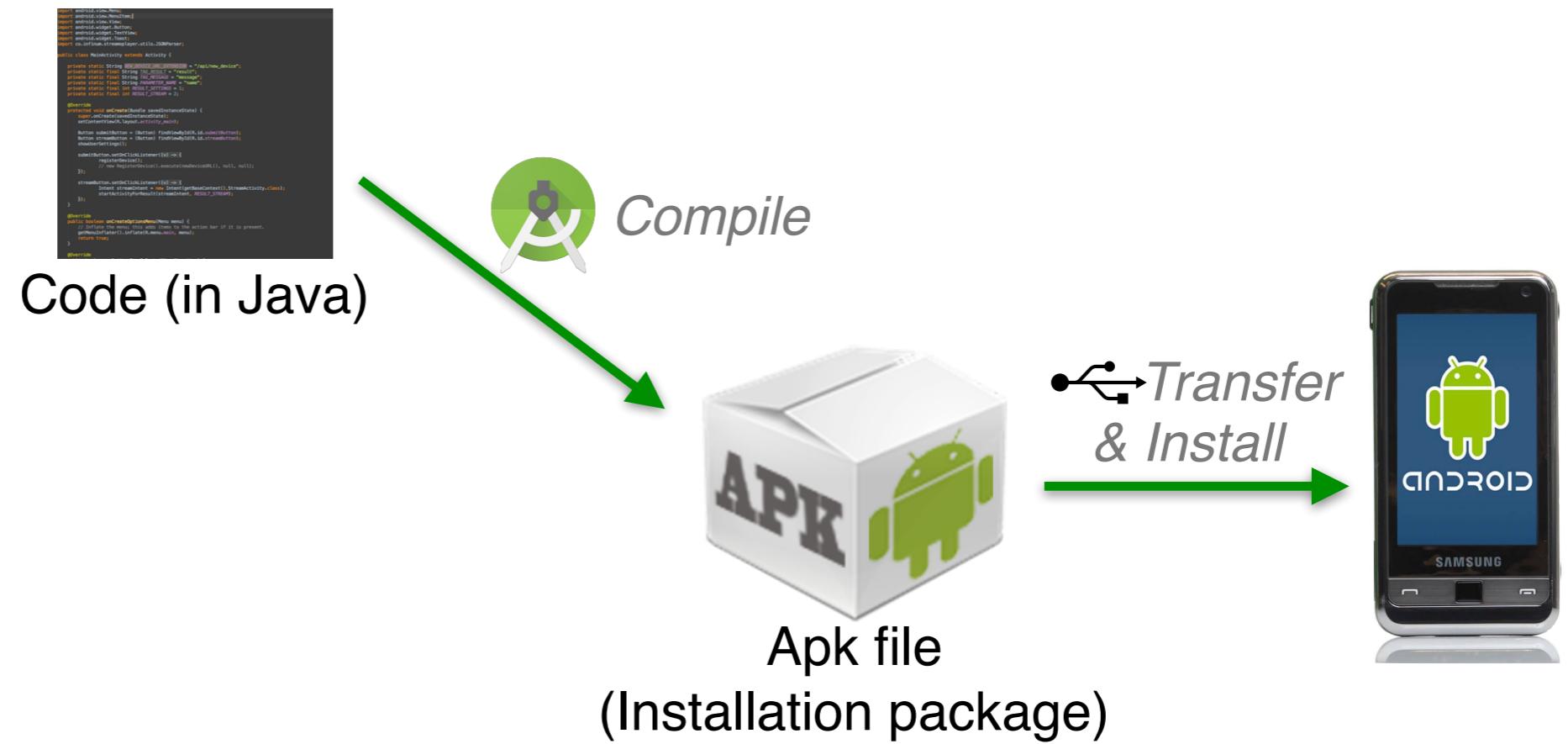
        streamButton.setOnClickListener((v) -> {
            Intent streamIntent = new Intent(getApplicationContext(), StreamActivity.class);
            startActivityForResult(streamIntent, RESULT_STREAM);
        });
    }

    @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()) {

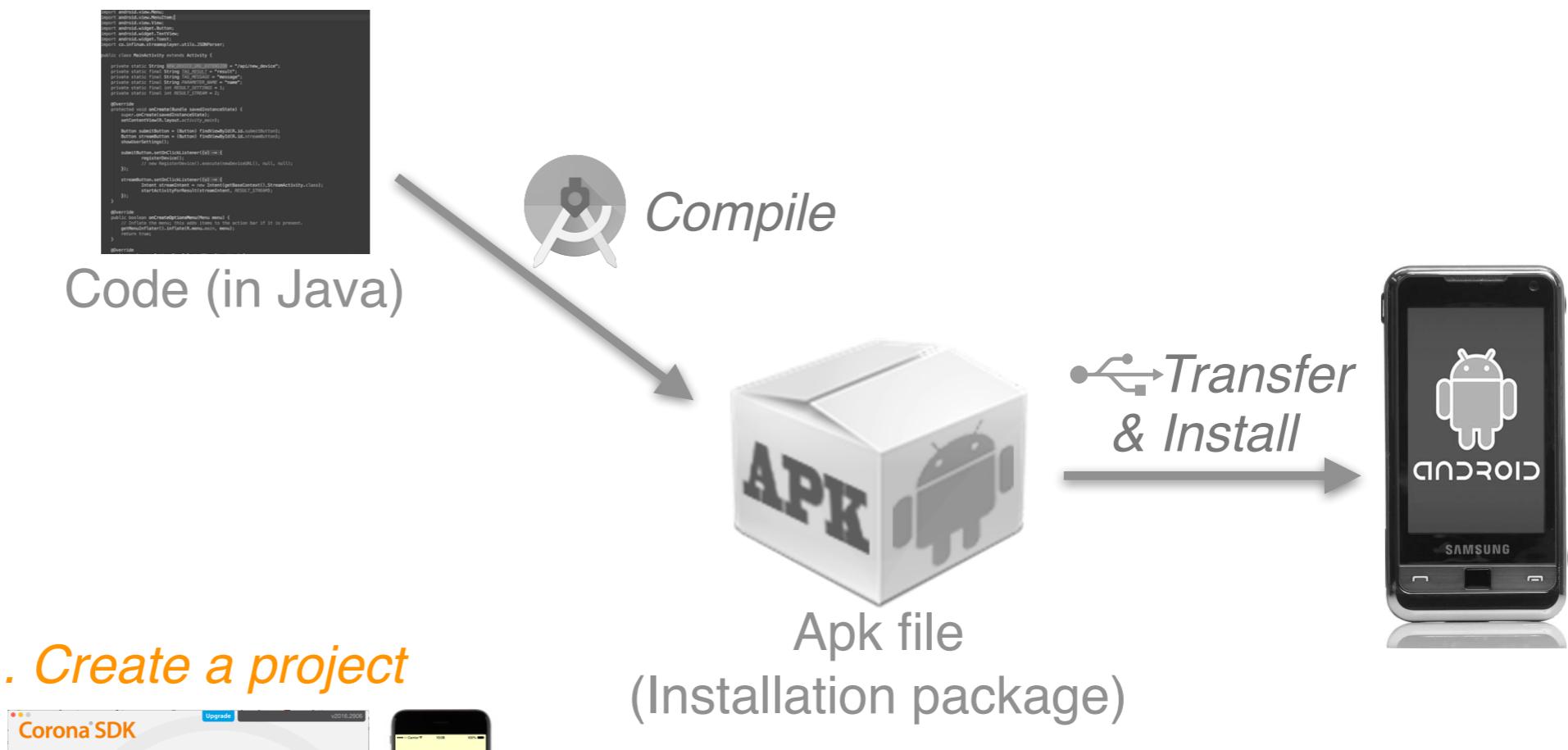
            case R.id.menu_settings:
                intent = new Intent(this, UserSettingsActivity.class);
                startActivityForResult(intent, RESULT_SETTINGS);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Compilation path of Android





Working flow in Corona SDK





Corona SDK

Working flow in Corona SDK

```
public static String TAG = "MyActivity";  
private static String URL = "http://www.device.com";  
private static String FILENAME = "result.html";  
private static final String PROPERTY_NAME = "name";  
private static final int REQUEST_CODE = 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);  
    streamButton.setOnClickListener(vl -> {  
        registerDevice();  
        Intent intent = new Intent(getApplicationContext(), StreamActivity.class);  
        startActivity(intent);  
    });  
    streamButton.setOnClickListener(vl -> {  
        Intent streamIntent = new Intent(getApplicationContext());  
        streamIntent.setAction("com.example.myapp");  
        startService(streamIntent);  
    });  
}  
  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.menu_main, menu);  
    return true;  
}
```

Code (in Java)



Compile



Apk file
(Installation package)

Transfer
& Install



2. Program in an editor using Lua

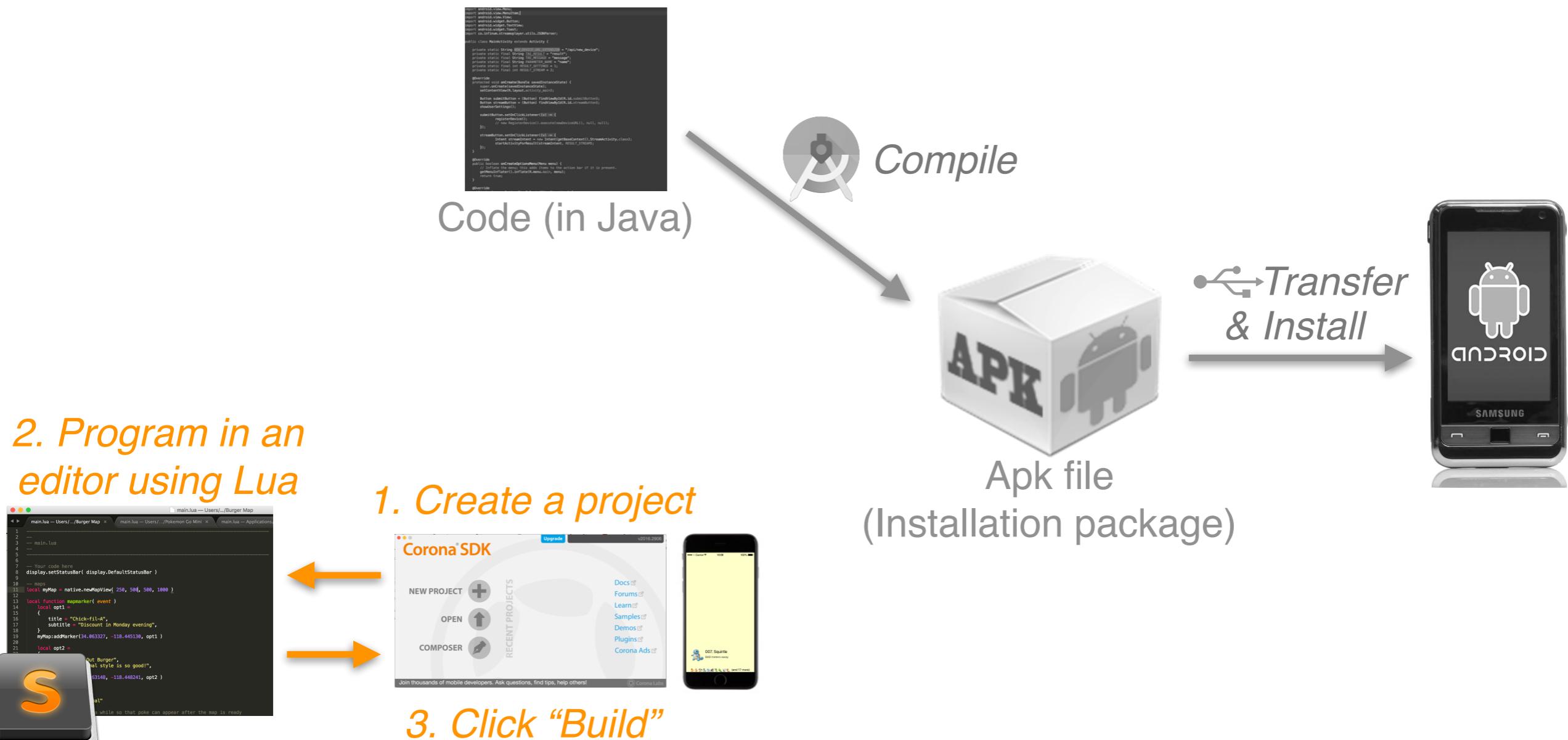
1. Create a project





Corona SDK

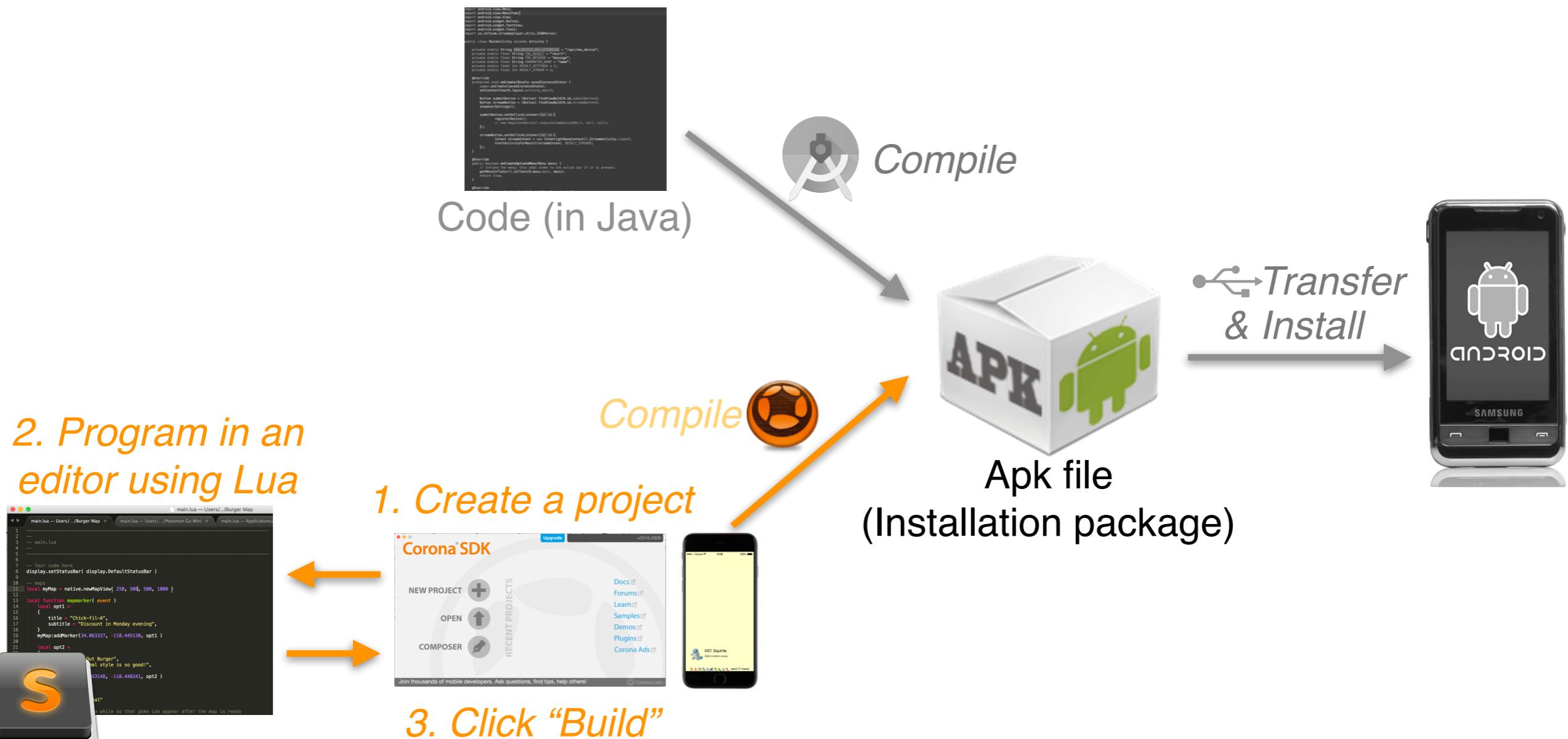
Working flow in Corona SDK





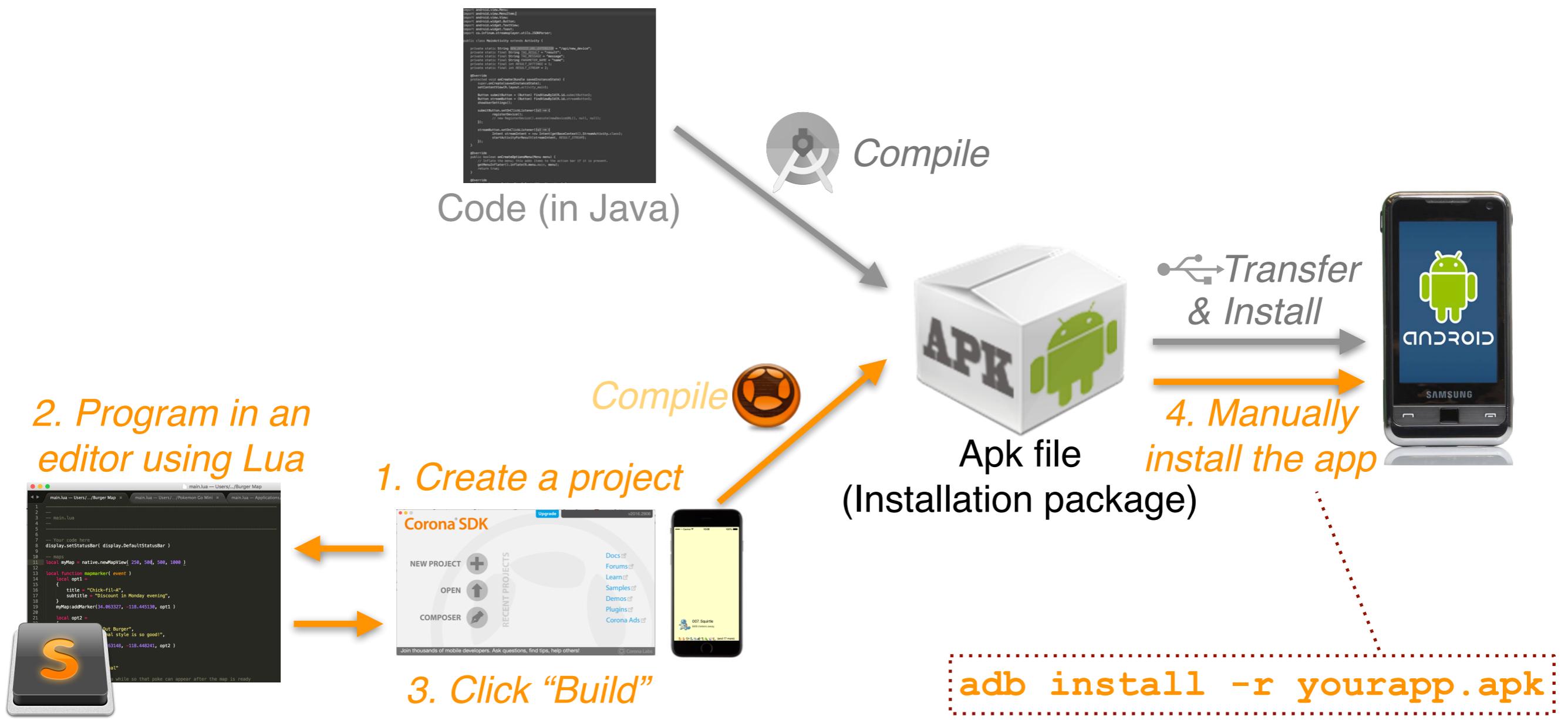
Corona SDK

Working flow in Corona SDK





Working flow in Corona SDK



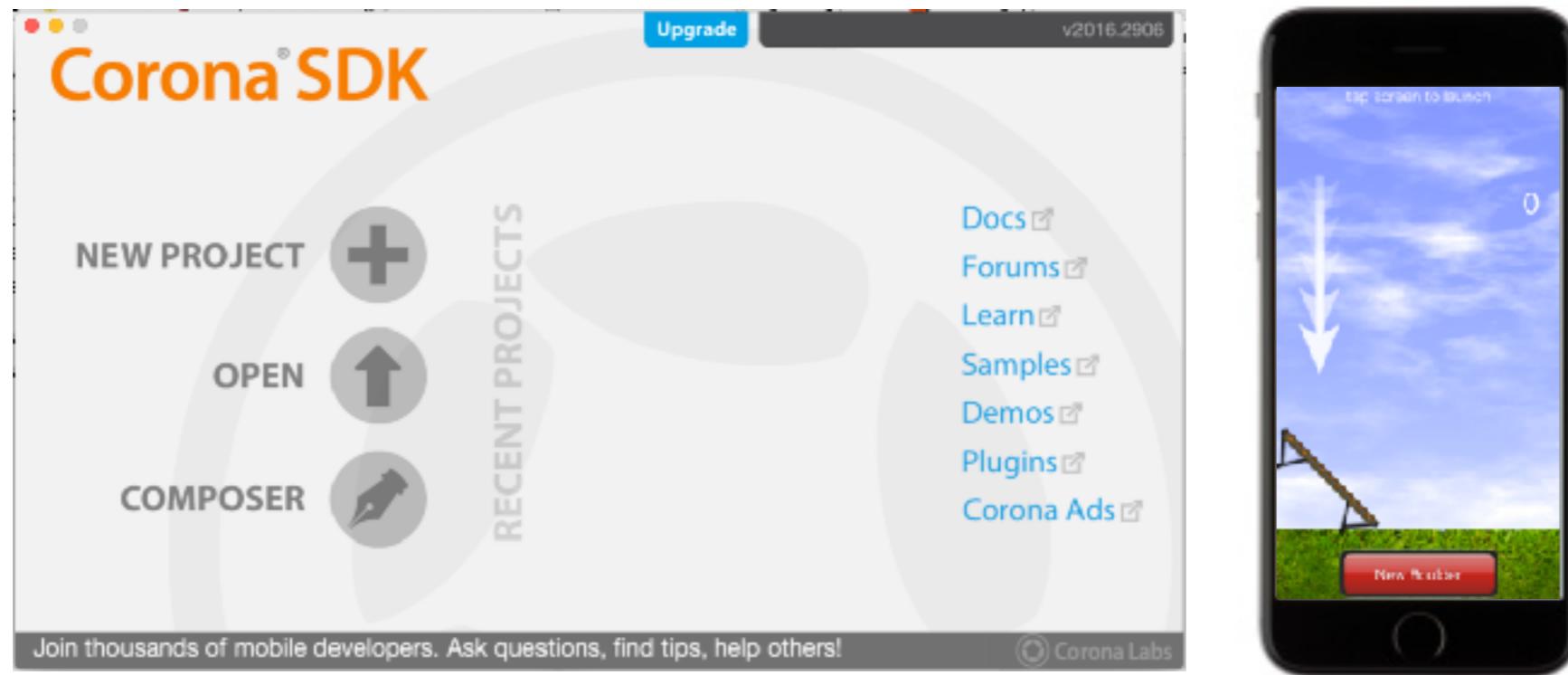
Episode 0 - There is a Gym...

Warm up



Project 00:

Run a sample program using Corona

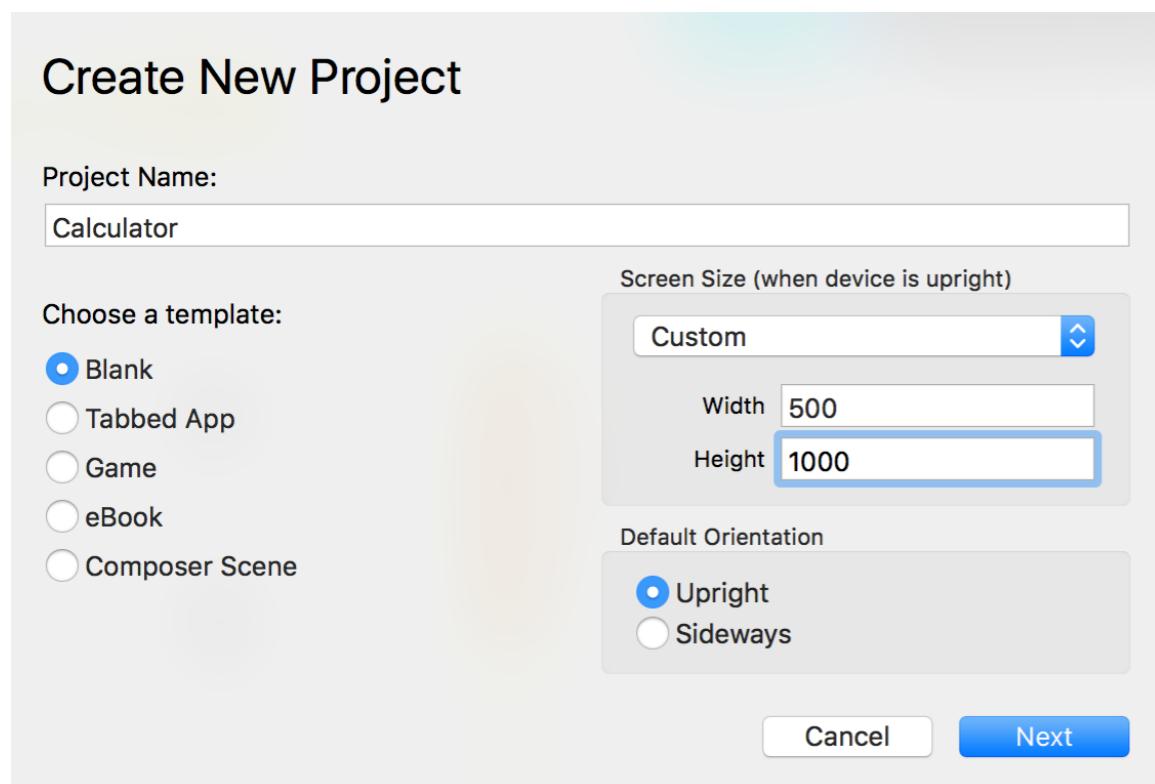


- Checkpoint 1: Be able to open Corona SDK
- Checkpoint 2: Build the app
- Checkpoint 3: Run the app on a phone

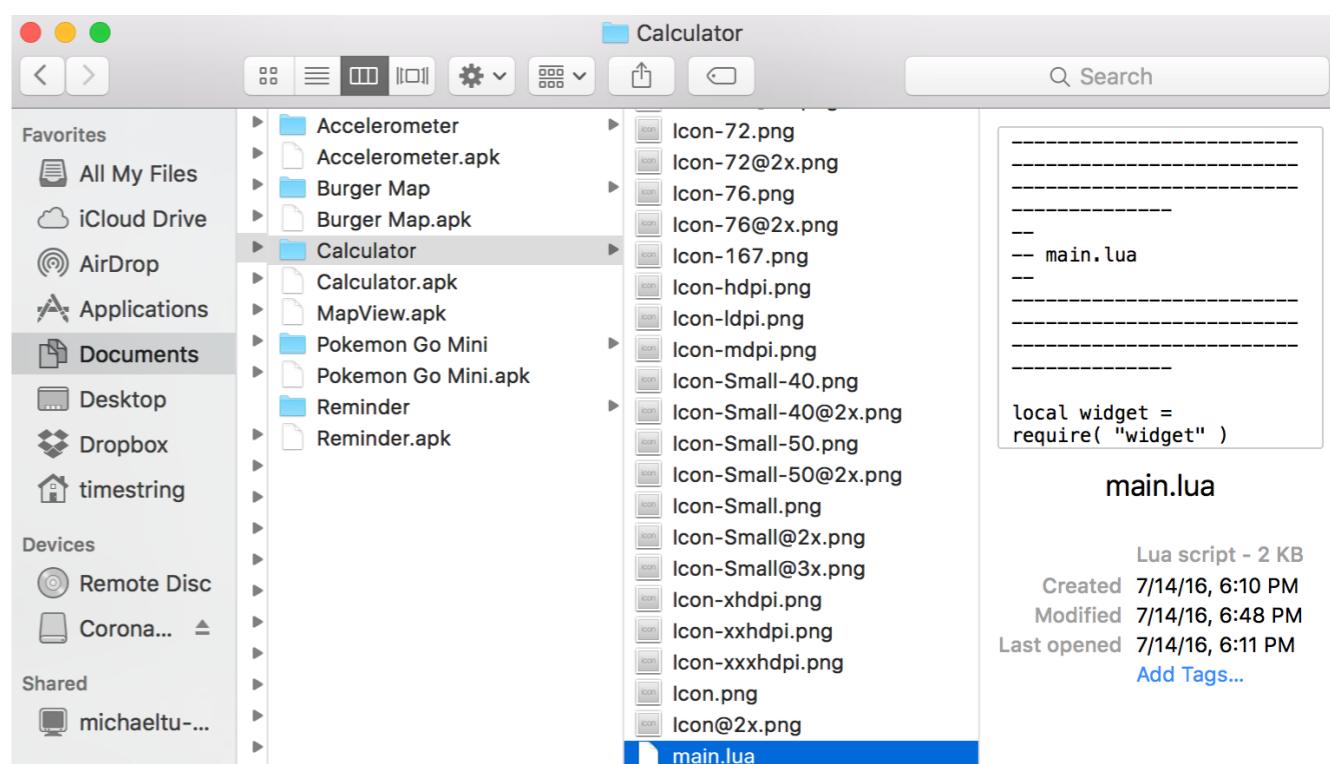
Project 00:

Run a Sample Program with Corona

- Step 1: Create a new project



- Step 2: Open main.lua in your editor



Introduction to Lua - general conventions

- Tip 1: Use variables
 - When the variable first time appear in your code, put keyword `local` in front of it (Don't ask!)
 - 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
```

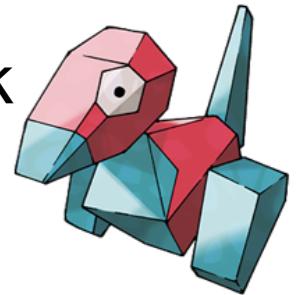


Expressions

- 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 do the trick



- Tip 6: Repetition - using loops

- Two versions, don't get confused

Start End

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

I get 10 numbers:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Start End End

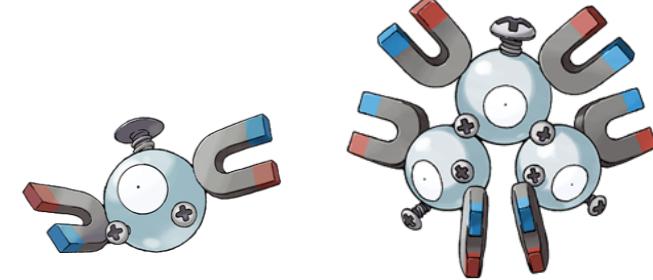
```
for i = 1, 10, 3 do
    print(i)
end
```

I get 4 numbers:
1, 4, 7, 10

Accessing attributes of objects

- Tip 7: Sometimes it's a dot, sometimes it's a colon!
 - If you want to become a master in this class, don't question!

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



- Tip 8: See sample codes
- Tip 9: Be confident! Ask the Instructor and TAs!
 - Search on Google, copy and paste, modify

Project 00:

Run a Sample Program with Corona

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

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

- Step 4: Say “Hello” to the world

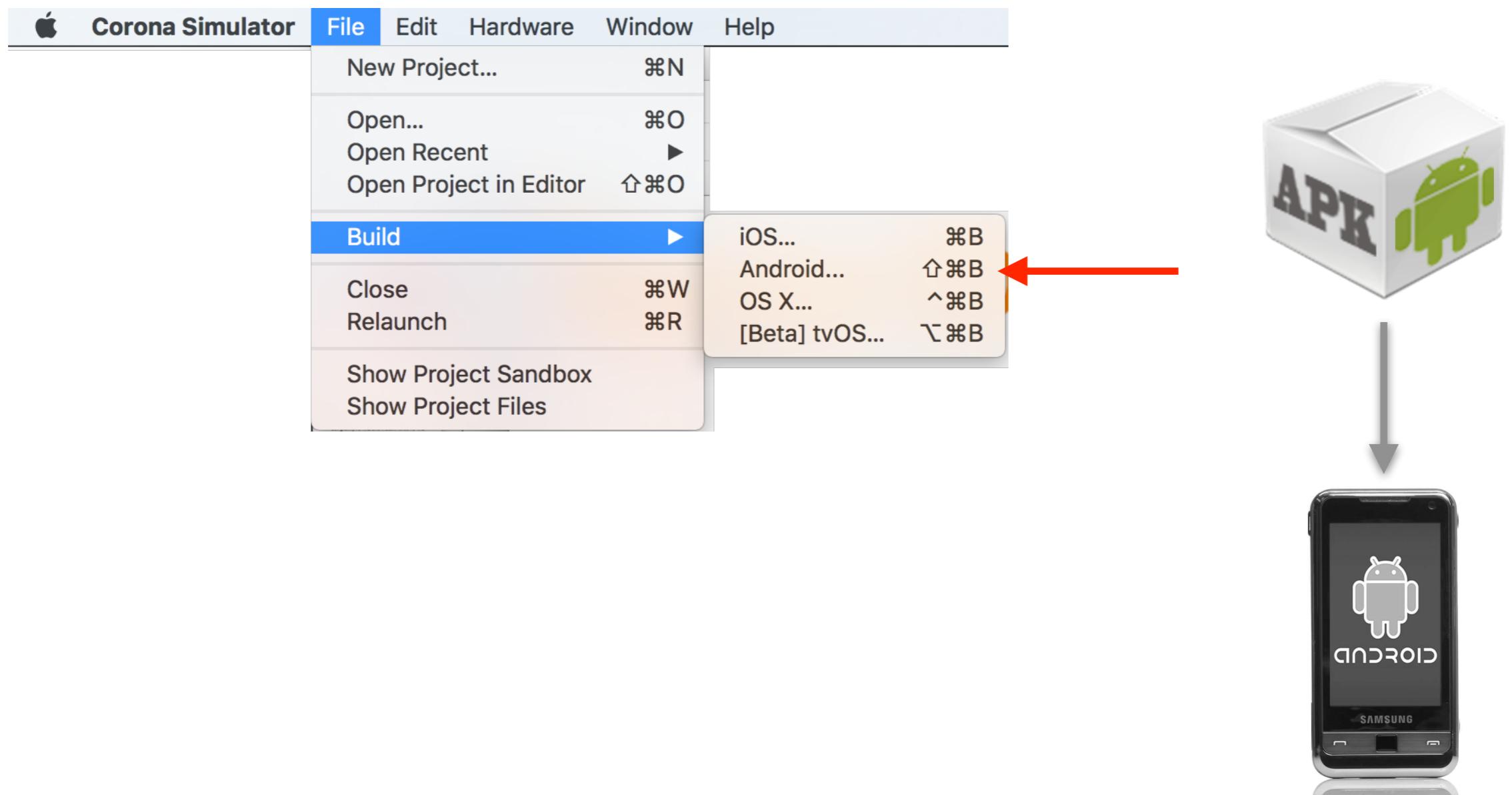
```
local helloWorld = display.newText("HelloWorld", display.contentCenterX, display.contentCenterY, native.systemFont, 32)
```



Project 00:

Run a Sample Program with Corona

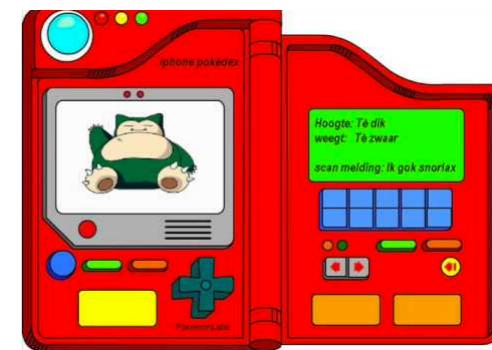
- Step 5: Build the app



Episode 1 - Pokedex

Calculator App

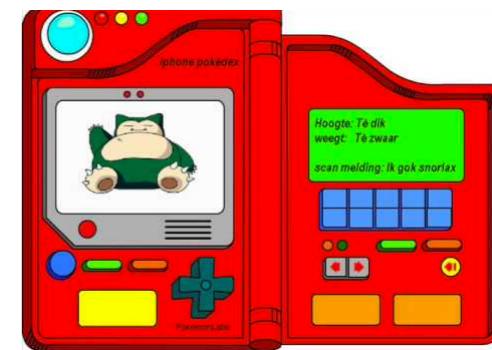




Project 01: Calculator App

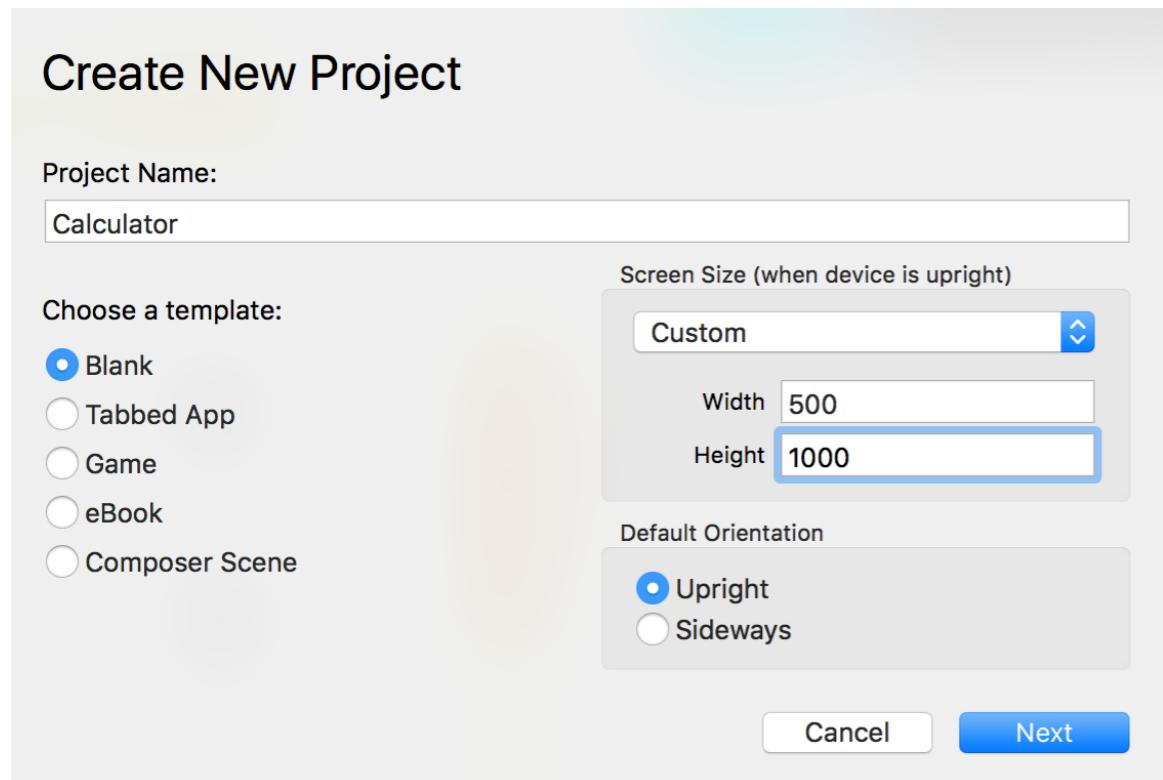


- Two blanks for users to enter numbers
- Arithmetic buttons
- Give results



Project 01: Calculator App

- Step 1: Create a new project



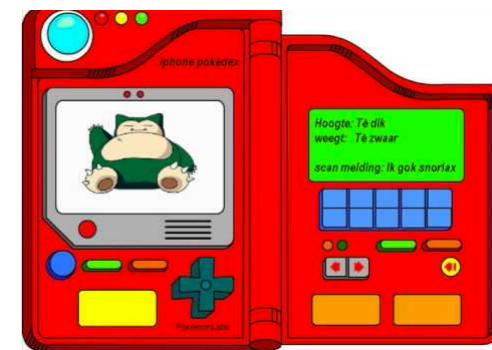
- Step 2: Open main.lua in your editor

```
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

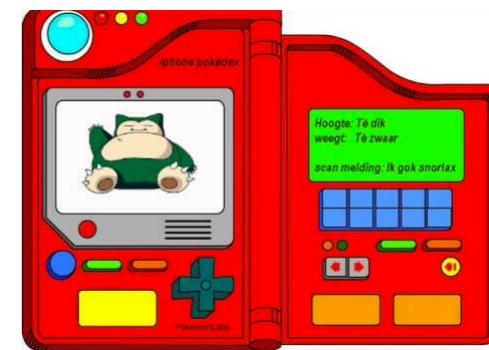
- Step 3: Always put this in the beginning of your code

```
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

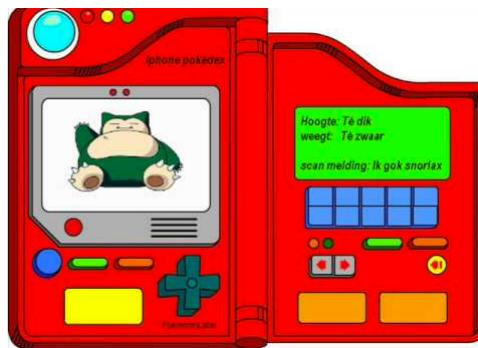
- Step 6: Put a text field to show the answer

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
```

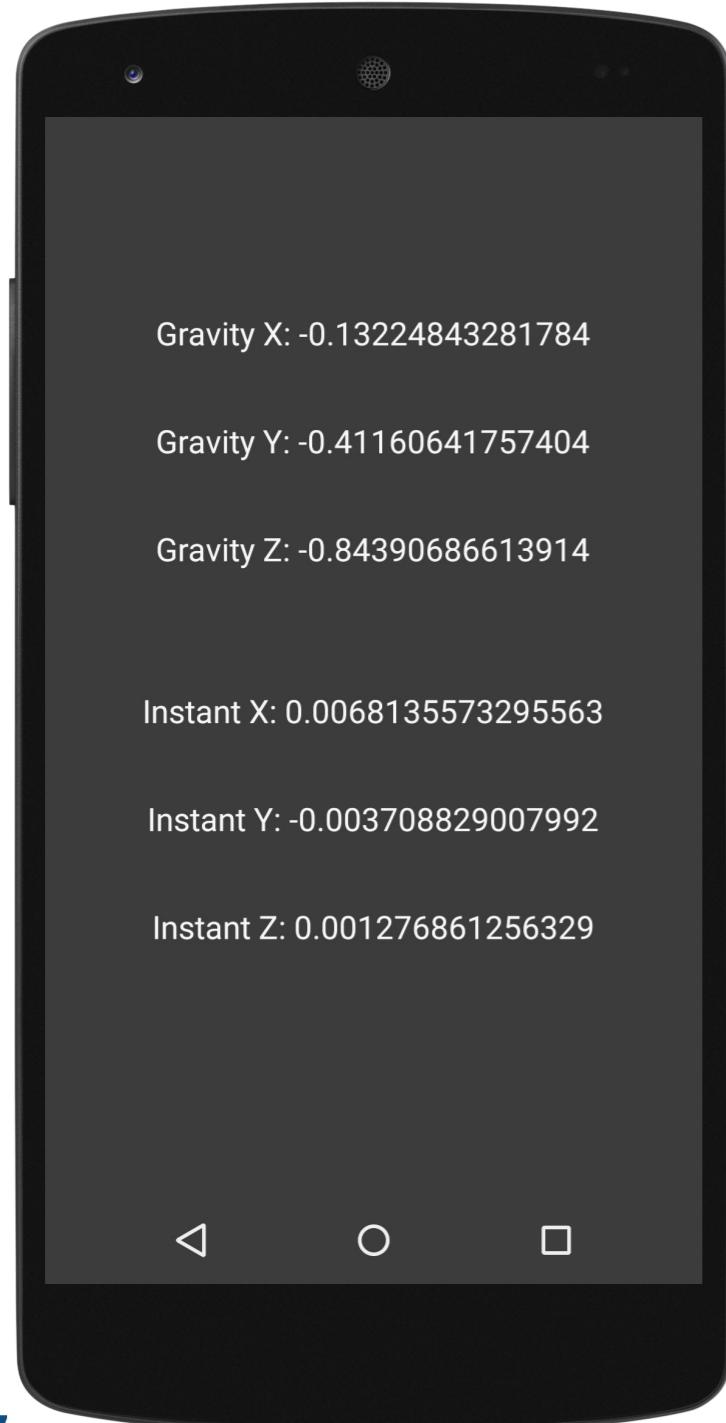
Episode 2 - How to Capture a Pokemon?

Accelerometer App





Project 02: Accelerometer App



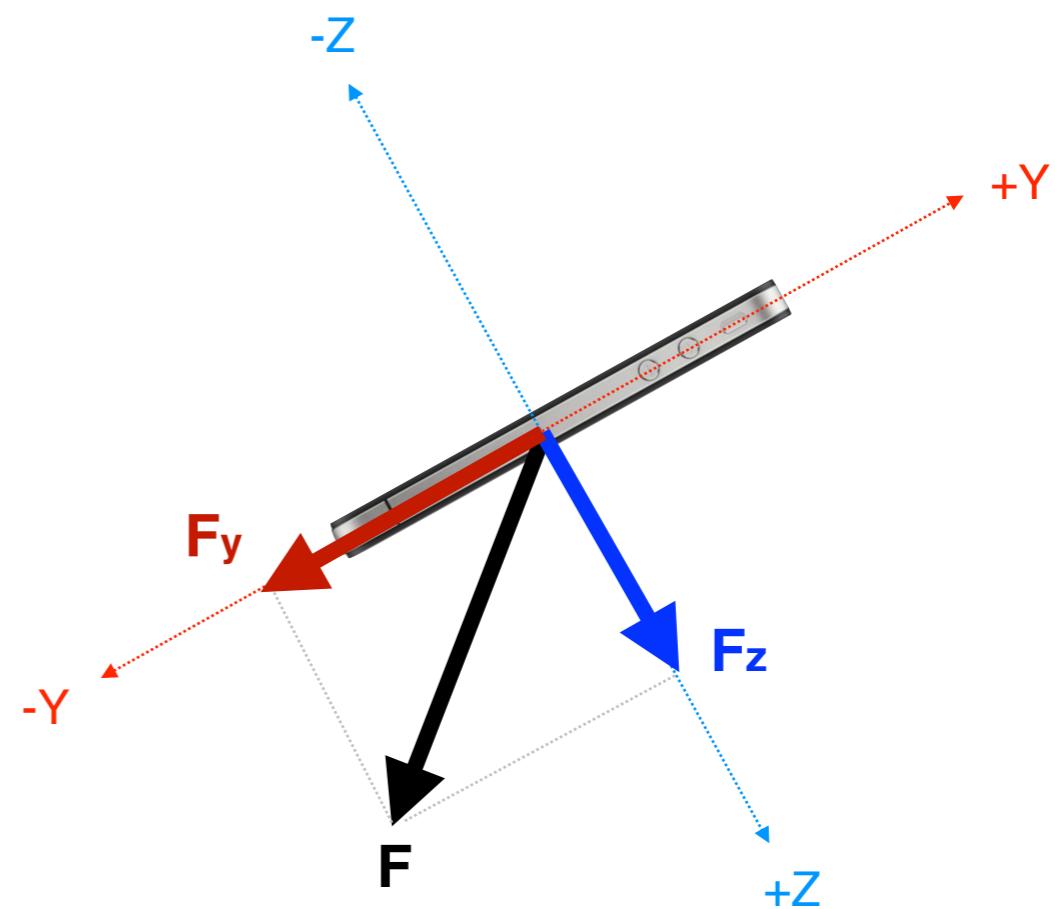
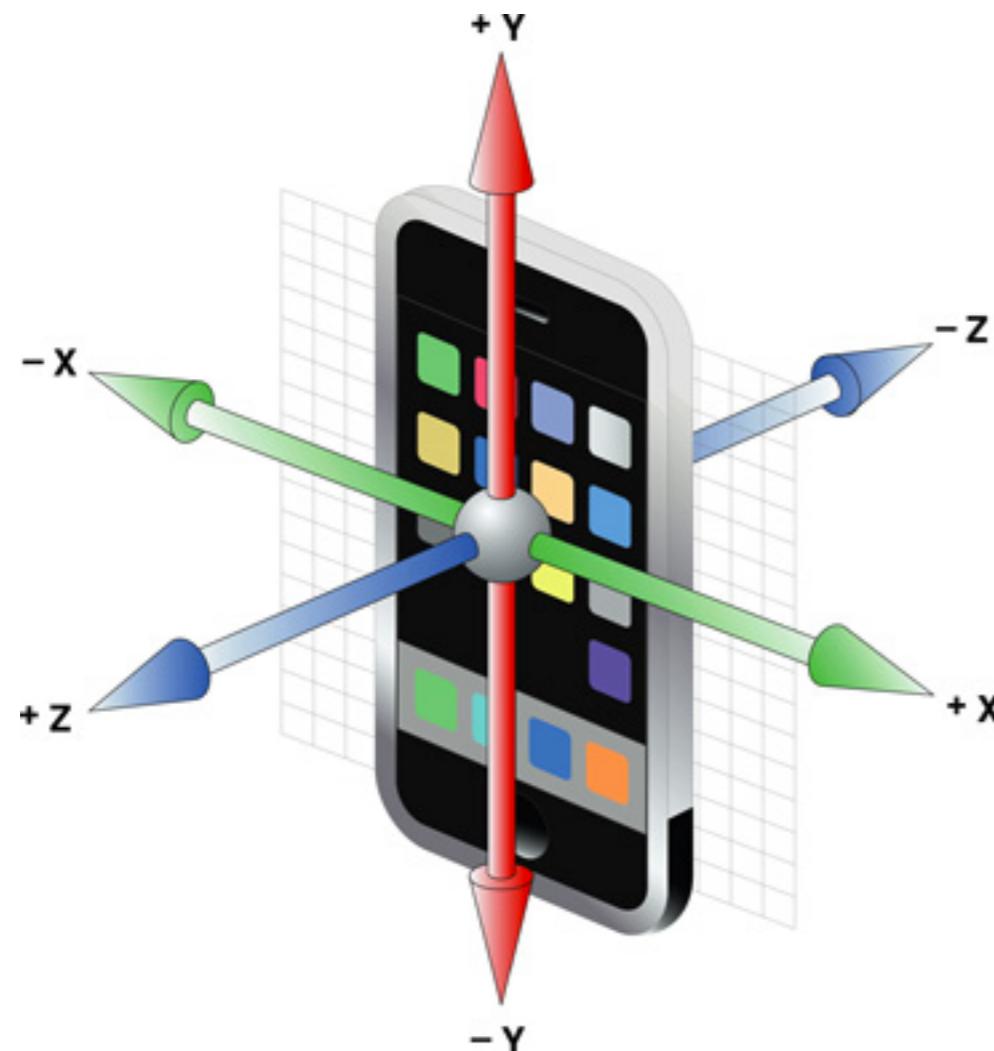
- Read values from accelerometer sensor
- Get gravity X, Y, Z and instant force X, Y, Z





Intro of accelerometer

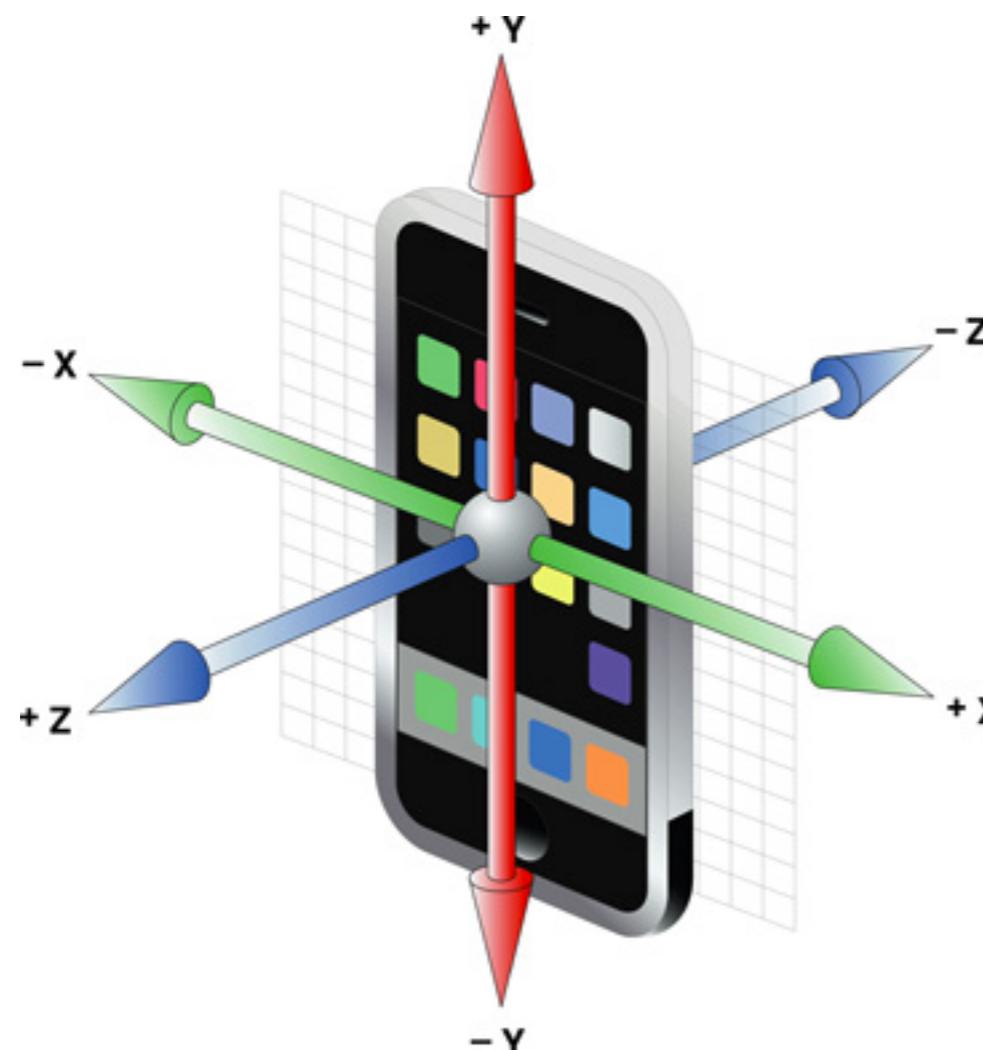
- Newton's First Law: $F = m * \underline{a}$



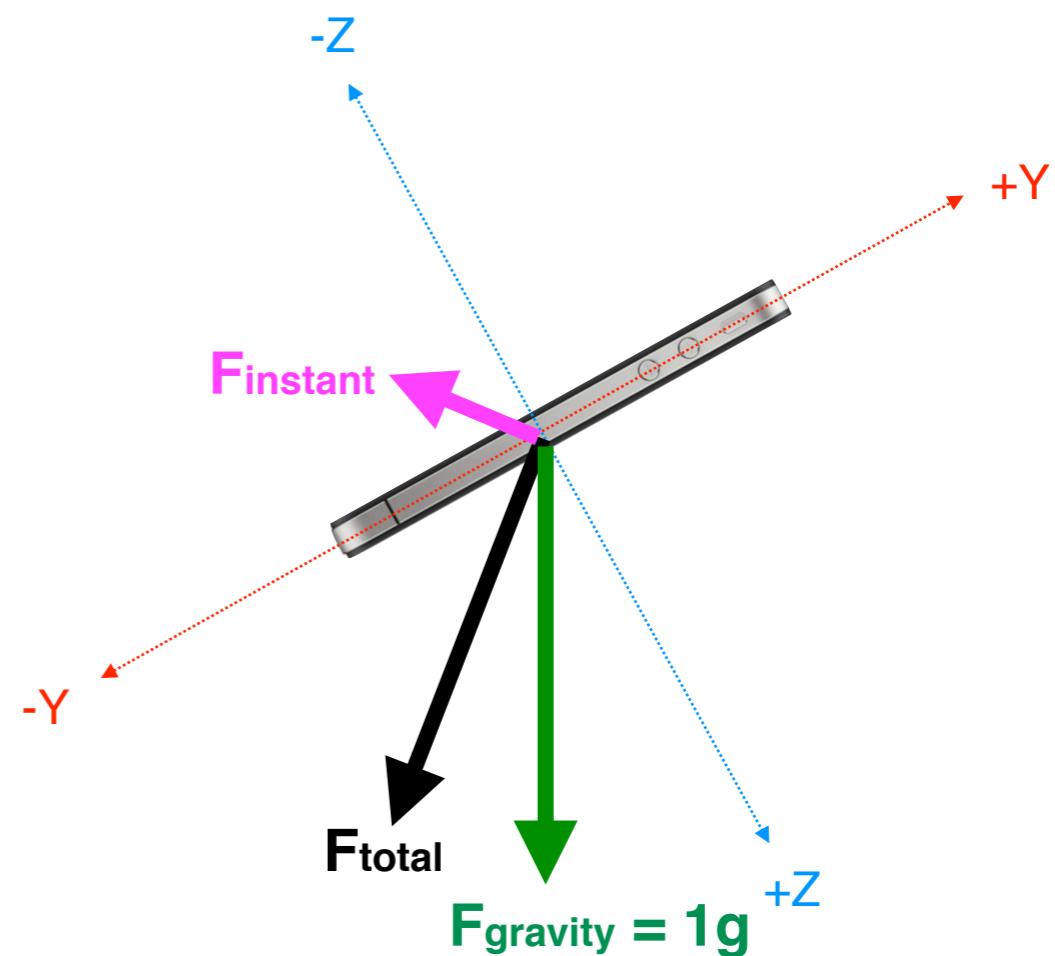


Intro of accelerometer

- Newton's First Law: $F = m * \underline{a}$



- Corona further helps us decompose to **gravity** and **instant** vector

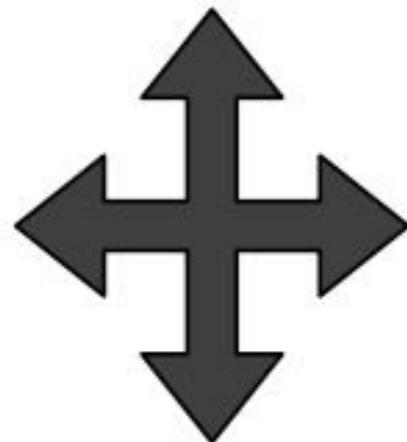




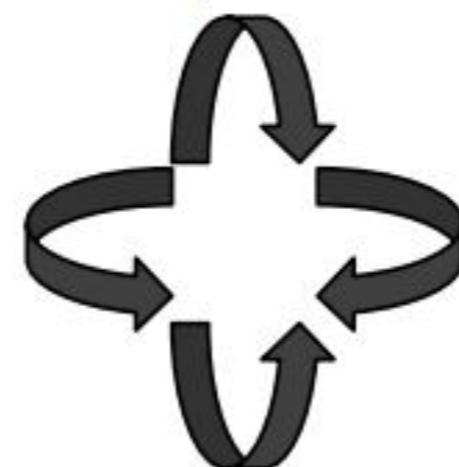
More sensors

- There are more hardware in phones, but we don't have time to explore them

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
```



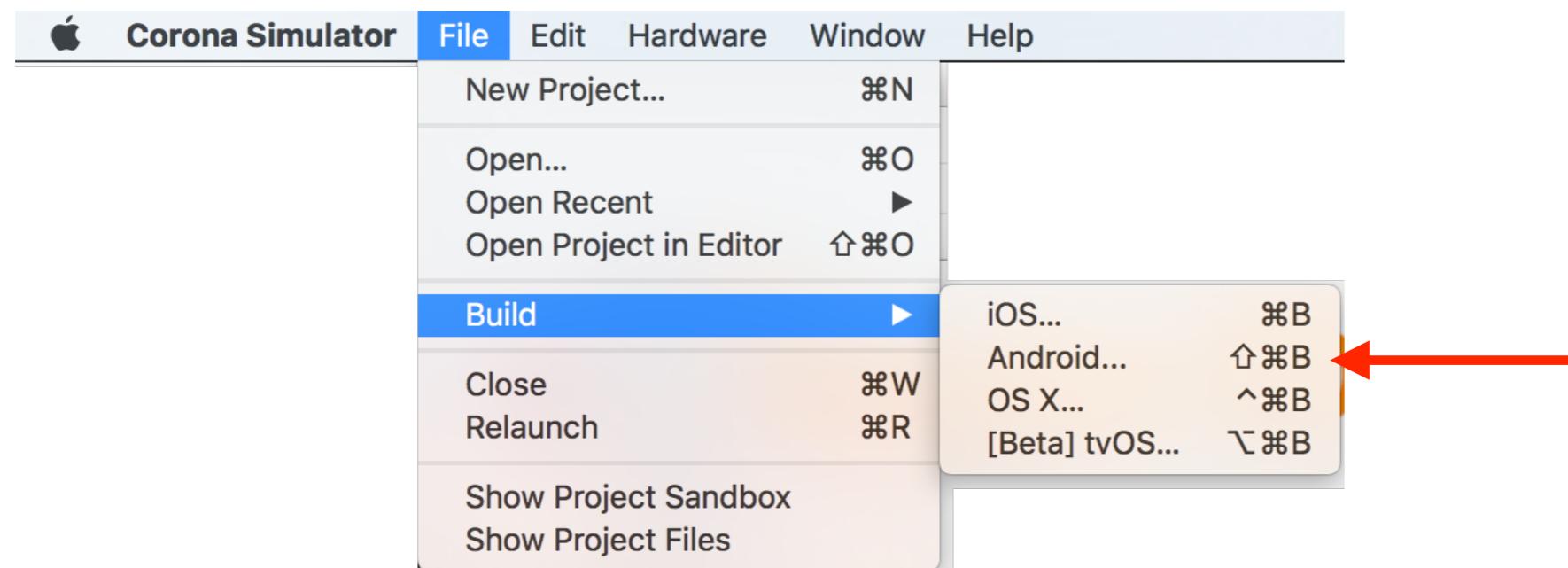
← Why leaving so much space?





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*)





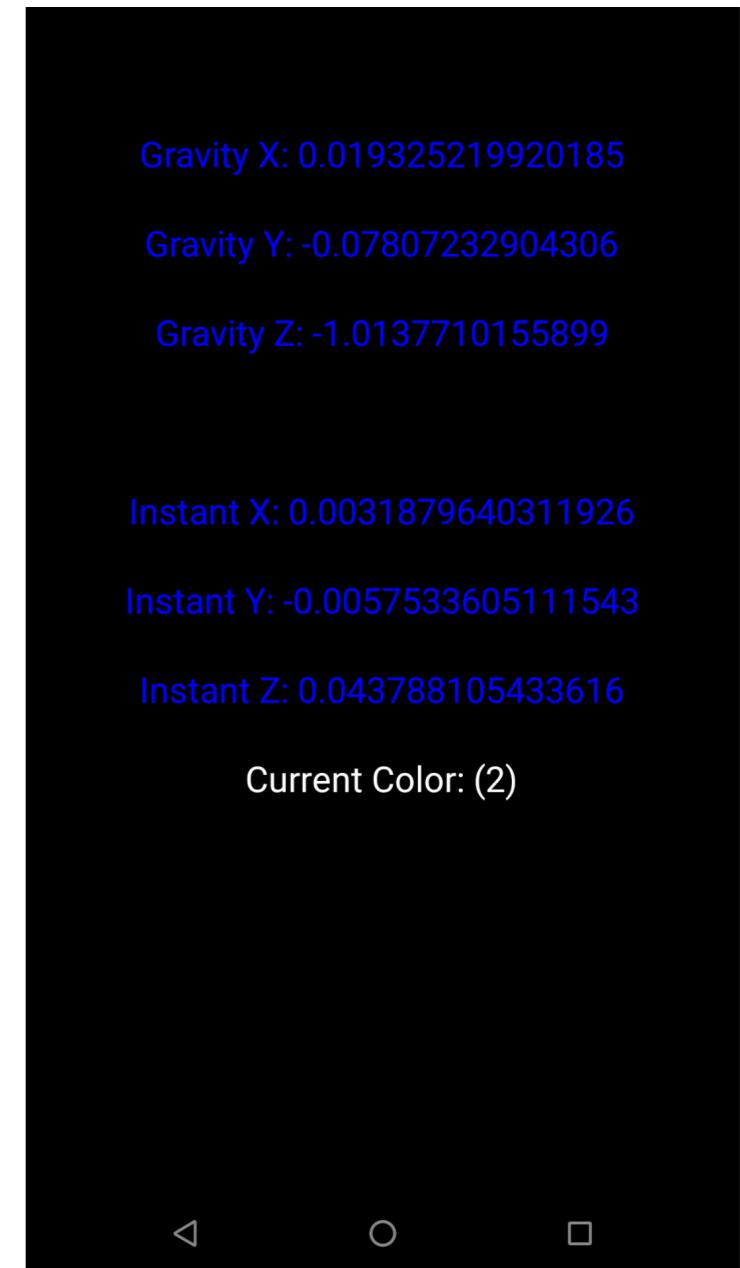
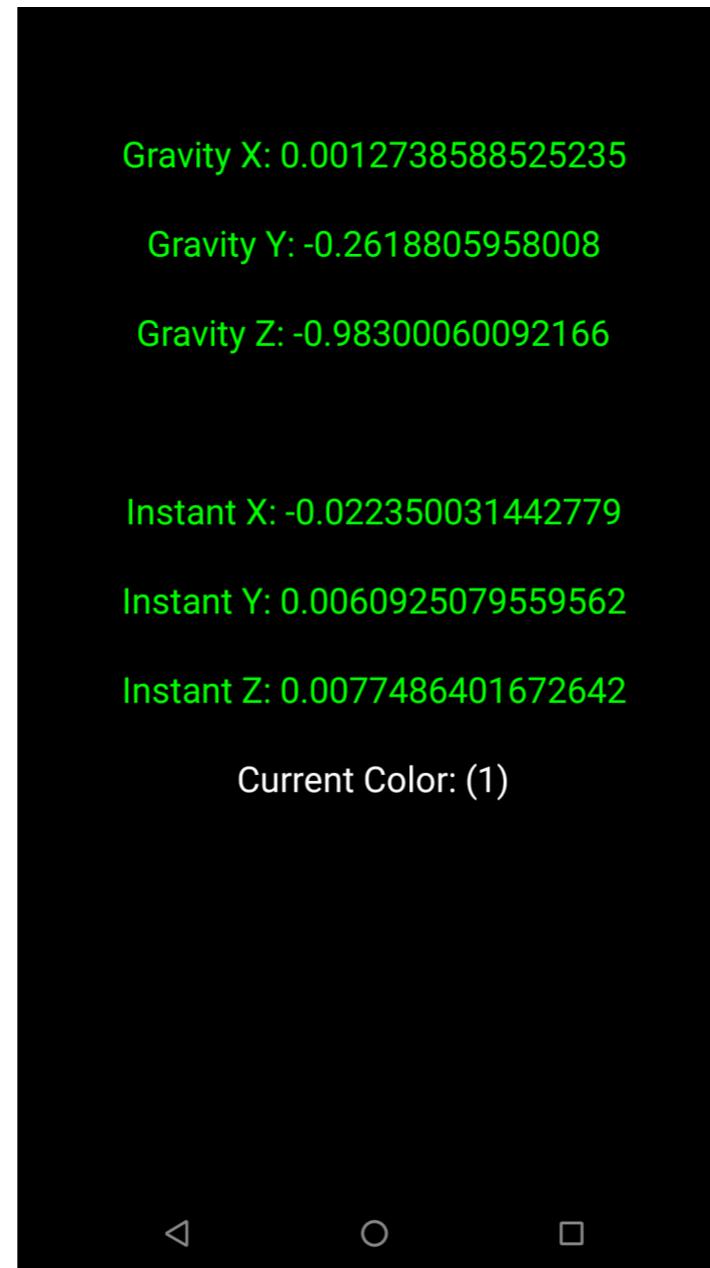
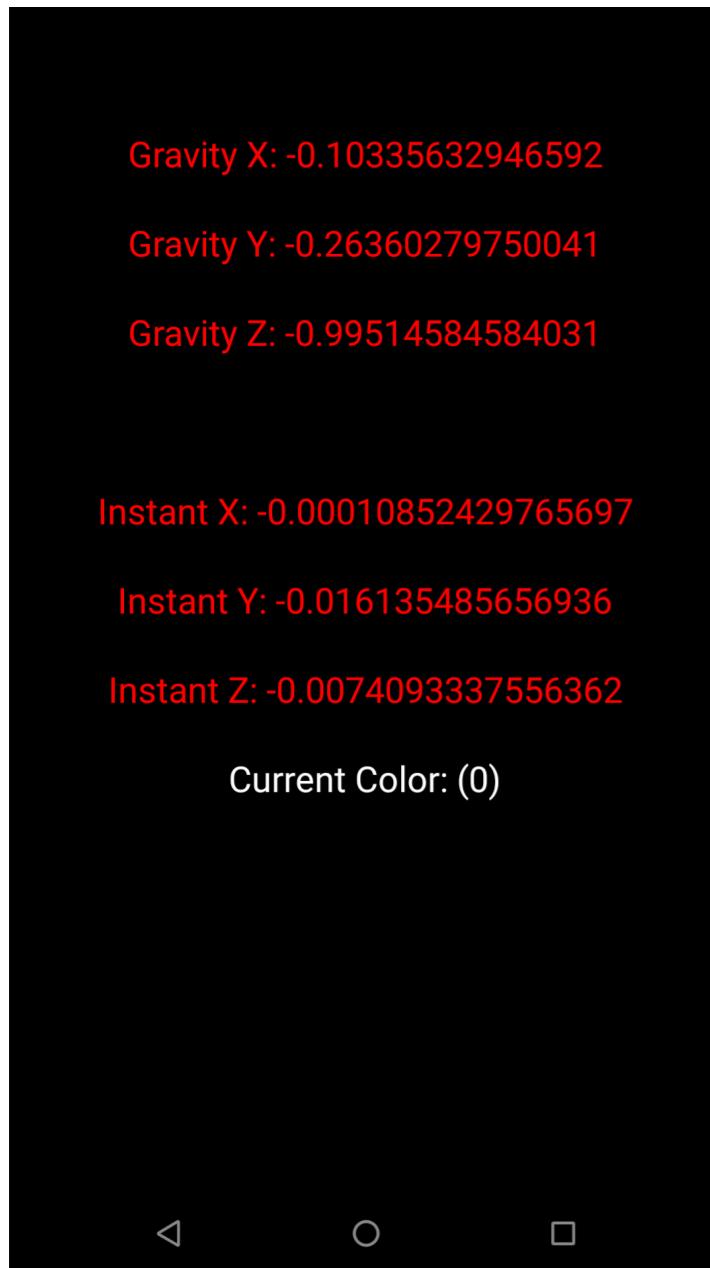
Project 02: Accelerometer App

- Secret step: Shaking detection! (shhhhh)

```
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
```



Project 02: Accelerometer App



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
```

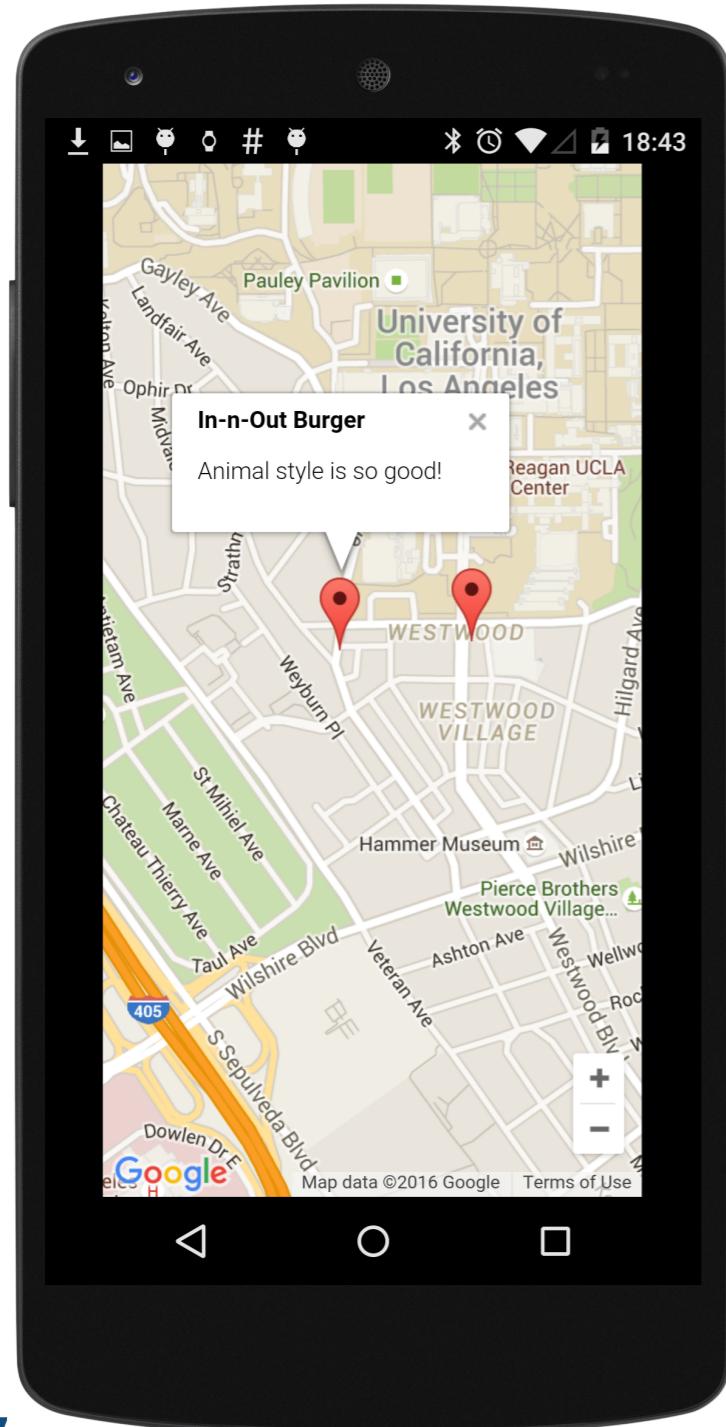
Episode 4 - Battle, I'm the Master!

Burger Map App





Project 04: Burger Map App



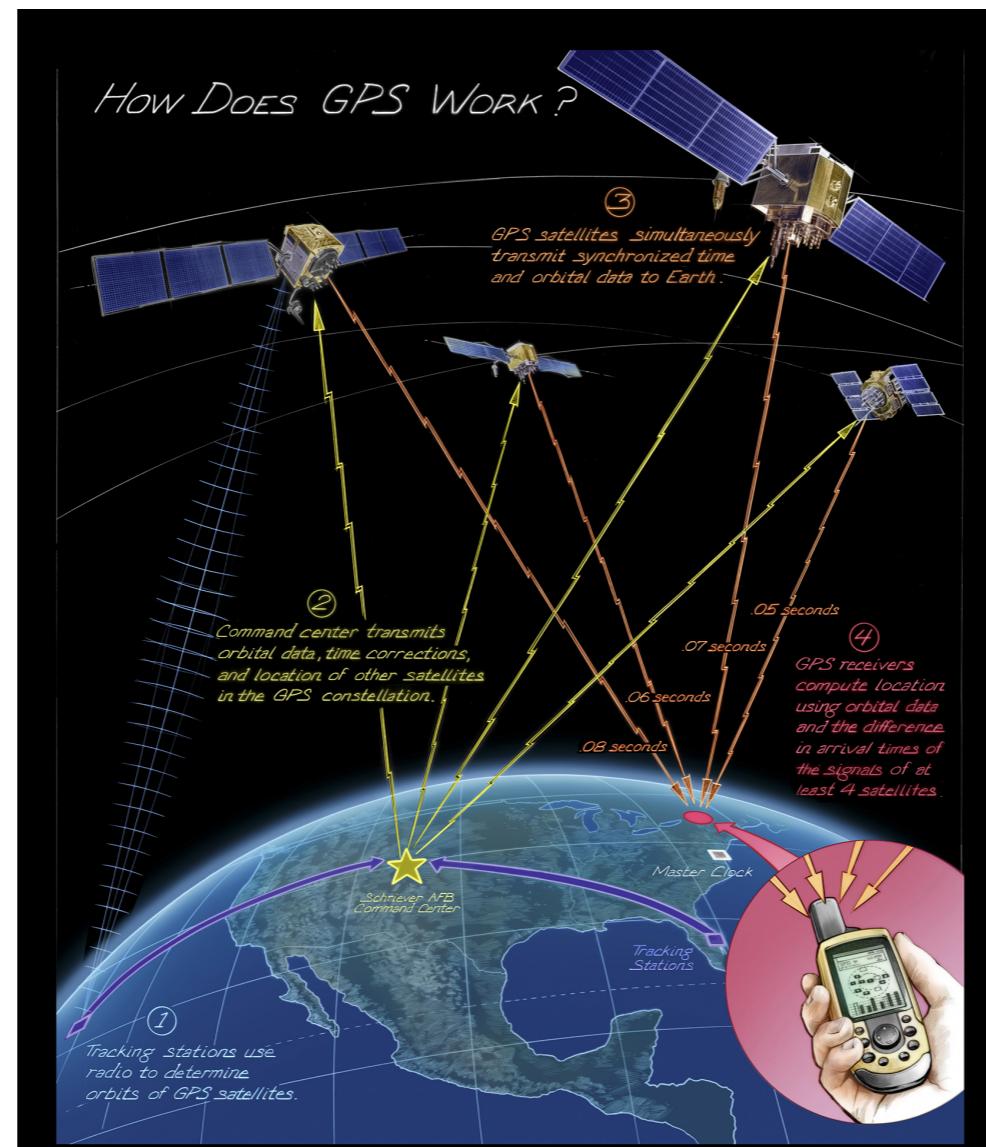
- Create a map app and select two places you want to introduce





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

Note: Make sure your phone is connected to WiFi

```
-- 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"

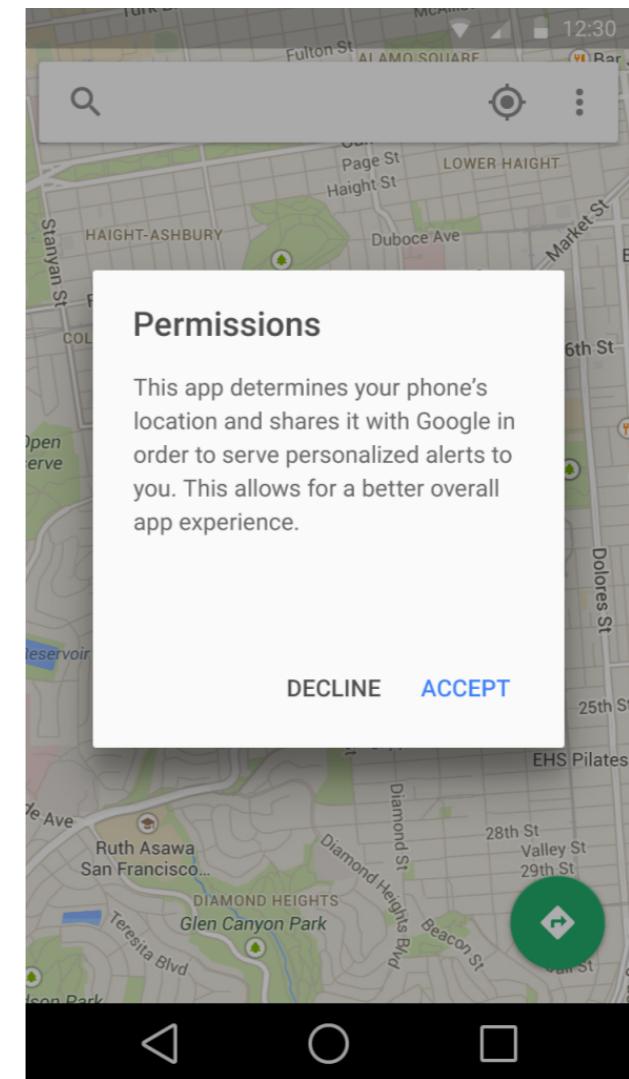
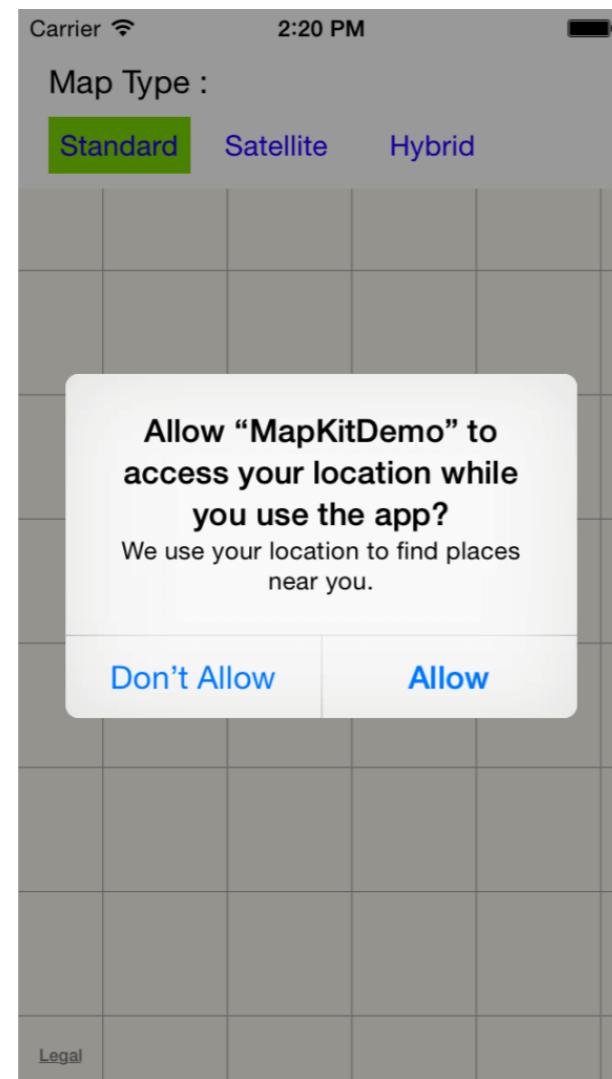
    -- need to delay for a while so that poke can appear after the map is ready
    timer.performWithDelay( 5000, mapmarker )
end
```





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

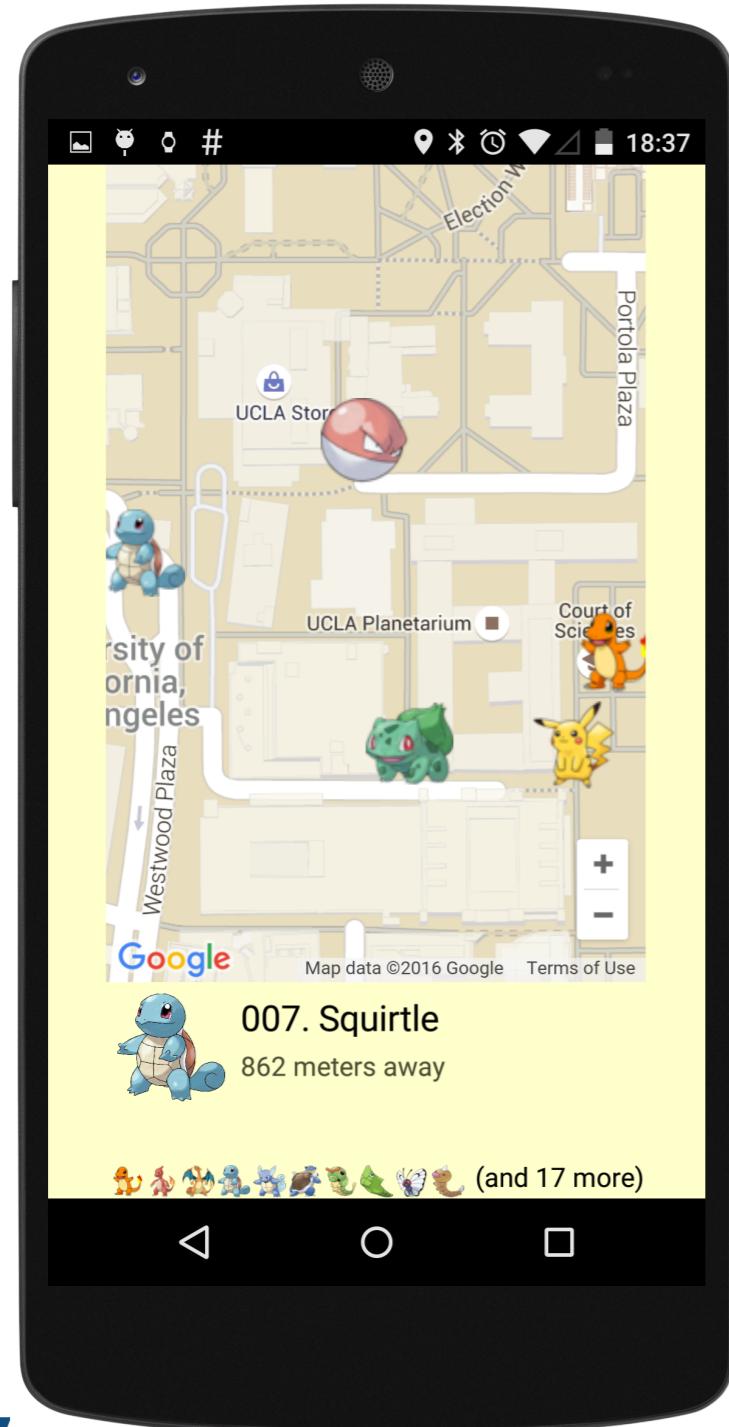
```
--  
-- Android Section  
--  
android =  
{  
    usesPermissions =  
    {  
        "android.permission.INTERNET",  
        -- Optional permission used to display current location via the GPS.  
        "android.permission.ACCESS_FINE_LOCATION",  
  
        -- Optional permission used to display 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

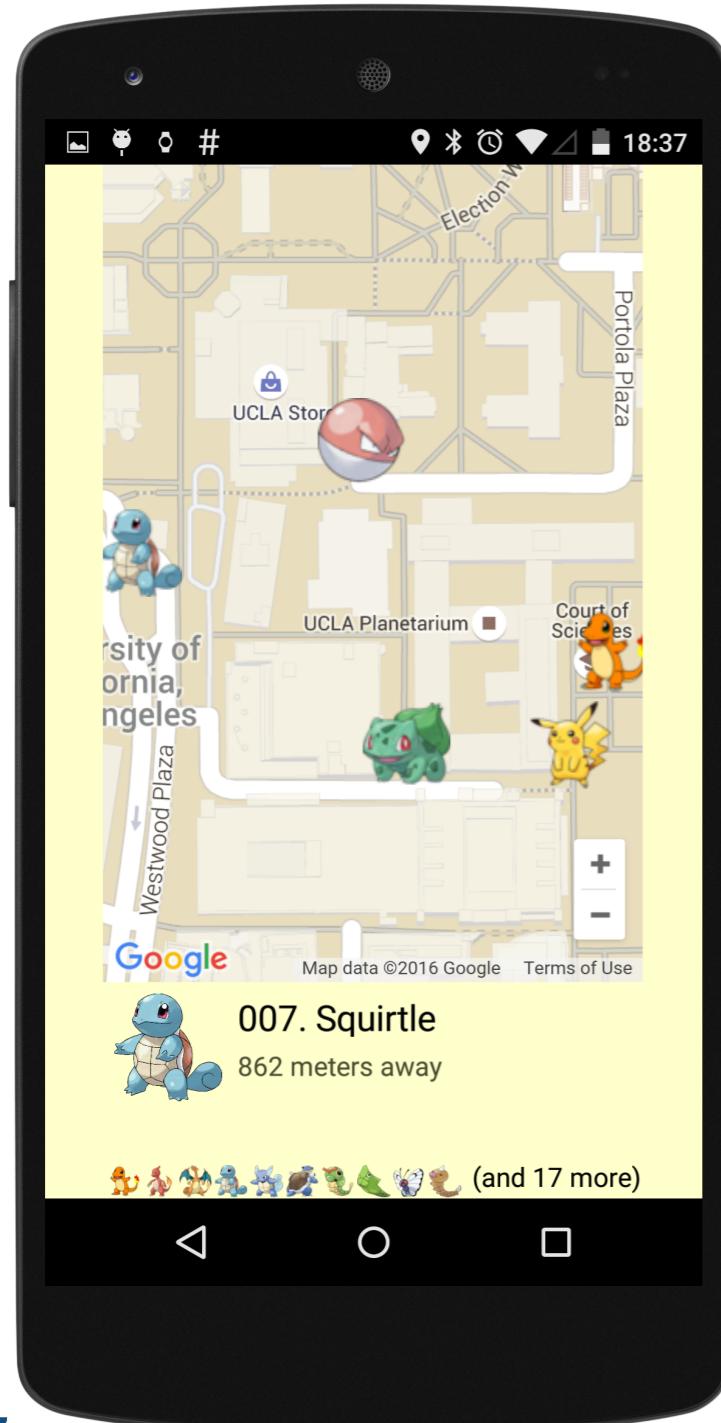


- Basic requirements
 - A map to show locations of Pokemon and yourself
 - Show distance between you and the nearest Pokemon
 - Detect Poke ball throwing action by using accelerometer





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
garcialuis@ucla.edu*