email: `egonzale@vub.ac.be`
office: 10F738

1 First steps in Android

Lab session material available at Pointcarre, and at `http://soft.vub.ac.be/amop/teaching/dmpp`
For information about Android, check out `developer.android.com` and `android-developers.blogspot.com`

1.1 Idea

The aim of this session is to get you familiar with the Android development environment. In particular, you will learn the Android basics such as working with Activities and intents, creating GUIs from xml by inflating them, working with AsyncTasks for enabling proper use of UI thread, etc. To do so, you will implement an Android version of the so popular memory skill game called *Simon*. The game consists of an Android device lighting up one or more buttons in a random order, after which a player must reproduce that order by pressing the buttons.

1.2 Implementing Simon

We implement Simon starting from a skeleton code in the Simon Android project that you need to import from the lab session material. As shown in Figure 1(a), Simon is an electronic device really popular in the 80's that had four colored buttons, each producing a particular sound when pressed by the player or activated by the device itself. When you run the Simon project, your device shows the main `Activity` (called `SimonActivity.java` in the project) with 4 buttons similarly to the original Simon device as shown in Figure 1(b).

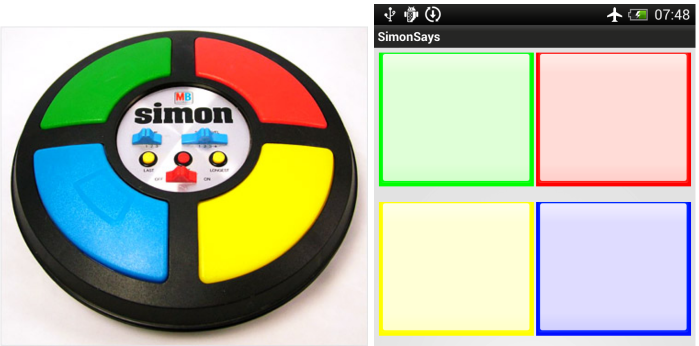


Figure 1: (a) The 80's Simon device (b) Main screen of Android Simon

A round in the game consists of the Android device lighting up one or more buttons in a random order, after which a player must reproduce that order by pressing the buttons. In the first round in the game, the Android device lights up one button, after which the player must press the button lighted up. At each round, the number of buttons lighted up increases, and as such, the number of buttons for the player to be pressed increases. In the Android project, the number of buttons to be pressed as the game progresses is implemented by the `Sequence.java` class. A `Sequence` keeps a vector of colors (representing the buttons to light up and then press), and an index indicating the progress of the player while reproducing the order given by Simon. At each round, the application extends the sequence with a new random color.

When you launch the provided Android Simon project with the skeleton code, the `SimonActivity.java` shows the main screen as shown in Figure 1(b), but Simon does not light up any sequence of colors; in other words, the `Sequence` is not played. Use the given skeleton code to incrementally grow the application as follows:

(a) Let's make Simon to play a sequence by completing the implementation of `play` method in the `Sequence.java` class. Note that in order to enable proper use of the UI thread, the play method spawns an `AsyncTask` and overrides the `doInBackground` tasks that looks over the current sequence and ask the UI thread to highlight each button and put it back to normal. To this end, the `SimonActivity` implements the `SimonGUI` interface which consists of a `highlightButton` and `normalButton` methods.

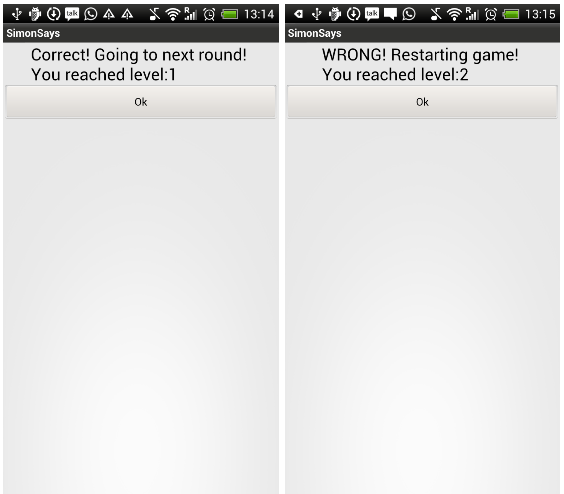
(b) Once Simon plays the sequence, it waits for the player to reproduce the sequence. Each time the player presses a button, the `butttonPressed` method in the `SimonActivity.java` class is called. As you can see in the skeleton code, the method first checks if the button pressed matches with the color in the sequence that Simon lighted up. When the player reaches the end of the sequence without making an error, the application launches a `ResultActivity` as shown in Figure 2 (a).

  (b1) Complete the code of `buttonPressed` method so that the application shows the `ResultActivity` when the player makes an error as shown in Figure 2(b).

  (b2) In order for Simon to show your `ResultActivity` activities, you first need to complete the GUI layout for this activity in the `activity_result.xml` file found under `/res/layout` folder in the Simon project, and then complete the `onCreate` method of the `ResultActivity.java` class to inflate such xml with the correct message string and level reached by the player.

(c) Congratulations, you have a basic Simon running! Let's have some extra fun ;). Add an option menu item where the player can chose the difficulty level similar to the original Simon which offered 4 difficulty levels. For example, in Simon the Rewind level asked you to repeat the sequence backwards.

(d) Let's pimp Simon! Add some tones when Simon plays the `Sequence` and/or some vibration effect when a button is pressed by the player.



Correct! Going to next round!
You reached level:1

WRONG! Restarting game!
You reached level:2

Figure 2: (a) Activity shown when giving correct sequence (b) Activity shown when an error is made

(e) Let’s go a step further! Add an option menu item for displaying the highest scores. To this end, add first a login activity in which the player can identify him/herself, so that you can show name of the user and highest score reached.

(f) Please rotate your screen and check that the game state is properly maintained. Check online information about how to do so using the `onSaveInstanceState` method.