# Lab 2: Tic-Tac-Toe in Android

Due: Feb 14th at 11:59pm

## Overview

In this lab, you will program a simple Android application to play Tic-Tac-Toe. The application consists of a 3x3 grid of buttons. The layout, buttons, and the `onClickListener()` is provided for you in the Android project. Your task is to complete the applications. Additionally, there are challenge and extra credit at the end of the document.

## Deliverables

Your submissions will include the entire Android proejct, but the files below are most important:

- `MainActivity.java`
- `READM`

Your `README` file must include, (1) your name, (2) the name of all files in your submission, and (3) a short description of the program and your development process. Failure to include a `README` or not including sufficient information in the `README` will adversely affect your grade on this lab.

## Project Retrieval and Submission Instructions

To retrieve the project for this lab, in Eclipse new project menu, choose a new project from SVN. Use the following URI and check out the project into the workspace:

```
https://cs71.cs.swarthmore.edu/svn/cs71/labs/02/TicTacToe
```

Once the project is checked out, you must then *disconnect* the project by right clicking on the project in the package explorer (Team→Disconnect). Also remove the SVN meta-data.

To submit the project, you will again right click on the project in the package explorer and choose Team→Share Project. Use the following URI as the svn repository location:

```
https://cs71.cs.swarthmore.edu/svn/cs71/submission/<username>/02
```

Where `<username>` is replaced with your swarthmore username. Click finish, and then follow the on-screen instructions to com mitt. You may update and commit your project until the deadline. All commits after the deadline will be rolled back to the submission just prior to the deadline.

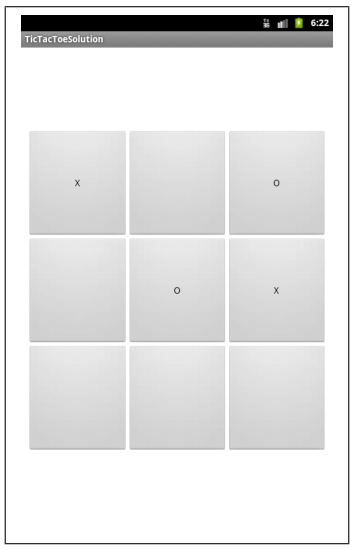


Figure 1: Sample gameplay of Tic-Tac-Toe Application

# TicTacToe

The game of Tic-Tac-Toe should be familiar to everyone. Briefly, the game consists of two players who alternate placing an X or an O, respectively, within a 3x3 grid. The players' goal is to connect three of their marks in a row, column, or in a diagonal before their opined does. If all grid squares are filled without a winner, the game is "Cats" or tied.

## Required Functionality

You are required to complete a functional game of Tic-Tac-Toe using the layout provided. The 3x3 grid consists of Buttons, and upon each press of the button either a X or an O will appear, alternating on each turn. After each turn, an end condition must be checked. A winner or cats will be declared if the end condition is met, or the game will continue until the end condition. See Figure 1 for a visual of the game play.

## Provide Code and Resource Files

The following JAVA and XML file are provided for your development.

### MainActivity.java

This is the main Java/Android program in your code. The code is well commented to direct your development. Below I highlight some key aspects of the code:

- `LOG_TAG` : This is a final string to use when using the LogCat logging facility. More information on this in the next section.
- `i_matrix[][]` : This is a 2-D matrix you will use to maintain the game state. A 0 indicates an X and 1 indicates a O, and a -1 indicates an empty square.

- `IntegerPair` : This is a private class defined within the MainActivity class that stores two integer pairs. You should use this class to store an index reference on the board, e.g., (0,0) is the uper left button, (1,1) is the middle button.
- `buttonOnClick(View v)` : This is the method that is called whenever a Button is pressed. The Button that is pressed is passed as the View v. You can cast that View to a Button.
- `toastWinnder()` and `toastCats()` : These two methods are called when there is a winner or a tie. The result should be a Toast message apearing on the screen. **You must figure out how to display toast messages on your own.** Google it!

**string.xml**

You are provided with three strings values set in `strings.xml`:

```
<string name="X">    X     </string>
<string name="O">    O     </string>
<string name="Blank">      </string>
```

Don't edit this file, but you should use these string values to set the text within the Button during game play.

**layout.xml**

The `layout.xml` file defines the layout of the buttons, see Figure 1 for the visual. Within the layout file, all the Buttons with IDs have been defined for you. Below is a sample of the XML file for reference, but you should review the rest of the layout file on your own for more details. You do not need to edit this file to complete this lab assignment.

```
 <LinearLayout
        android:gravity="center_horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="horizontal">

        <Button
            android:id="@+id/button_0_0"
            android:layout_height="fill_parent"
            android:layout_width="fill_parent"
            android:layout_weight="1"
            android:onClick="buttonOnClick"
            android:text="@string/Blank"/>
        <Button
            android:id="@+id/button_0_1"
            android:layout_height="fill_parent"
            android:layout_width="fill_parent"
            android:layout_weight="1"
            android:onClick="buttonOnClick"
            android:text="@string/Blank"/>

        <Button
            android:id="@+id/button_0_2"
            android:layout_height="fill_parent"
            android:layout_width="fill_parent"
            android:layout_weight="1"
            android:onClick="buttonOnClick"
            android:text="@string/Blank"/>

    </LinearLayout>
```

Above is the XML definition for the first row of Buttons. You'll notice that the `android:id` value provide the index: The first row is 0 and each button 0 through 2. Also notice that the `andorid:onClick` is set to `buttonOnClick` (see above), which is the method that will be called when the button is pressed. **Upon the click for all Buttons, the same method is called. One challenge in this assignment is determining which Button is clicked.** Hint: You can initialize a data structure in the `onClick()` method to make this process much easier.

### Debugging Using LogCat

Although it is not required, I strongly recommend you use Android's logging facility to aid your debugging. Essentially, this is a way to print out information about the state of the game. The result of these log messages will appear in the the LogCat messages in Eclipse, or you can access them using `adb logcat`.

To log a message, use the following code for information logs:

```
Log.i(this.LOG_TAG, <string>)
```

And the following code for debug loggs:

```
Log.d(this.LOG_TAG, <string>)
```

To aid your debuggin, I've provided a method to convert the `i_matrix` to a nice visual: `matrixToString()`. For example, here is some LogCat output from Eclipse upon running my solution code:

```
02-02 18:16:59.885: I/Tic-Tac-Toe(367): Button Clicked turn:O index: (1,1)
02-02 18:16:59.896: I/Tic-Tac-Toe(367): End Condition Checking:
02-02 18:16:59.896: I/Tic-Tac-Toe(367): X _ O
02-02 18:16:59.896: I/Tic-Tac-Toe(367): _ O X
02-02 18:16:59.896: I/Tic-Tac-Toe(367): _ _ _
```

You can read more about the logging facility in *Android Programming* and on-line in the Android documentation.

## Challenge Exercises and Extra Credit

- (**Challenge**) Update the text and font size for the buttons so that it the text is nice and big and bold. Also, have X appear in blue and O appear in red.
- (**.5 Points**) Add an extra view to the layout at the top, a `TextView`, that tracks the number of games won by each player.
- (**1.5 Points**) Program Super Tic-Tac-Toe that requires a grid of 4x4, but a winner is still determined by getting three X's or O's in a row. You'll need to change the layout file. To submit Super Tic-Tac-Toe, you will check in an additional project into the SVN submission repository name SuperTicTacToe. Also, indicate this in your README file.