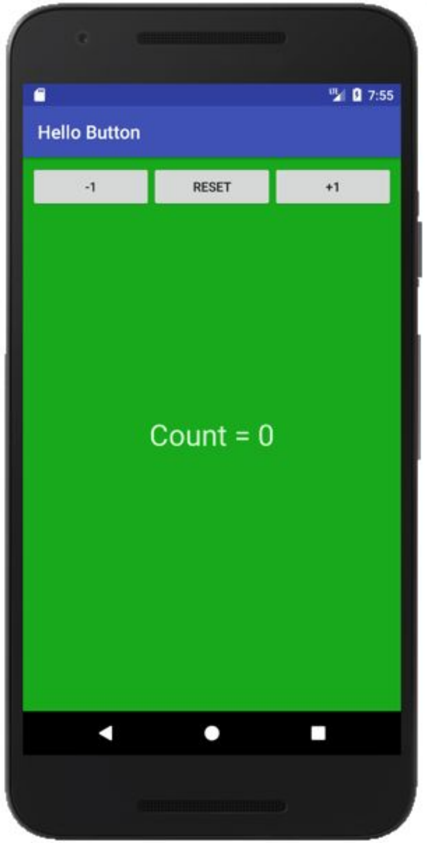
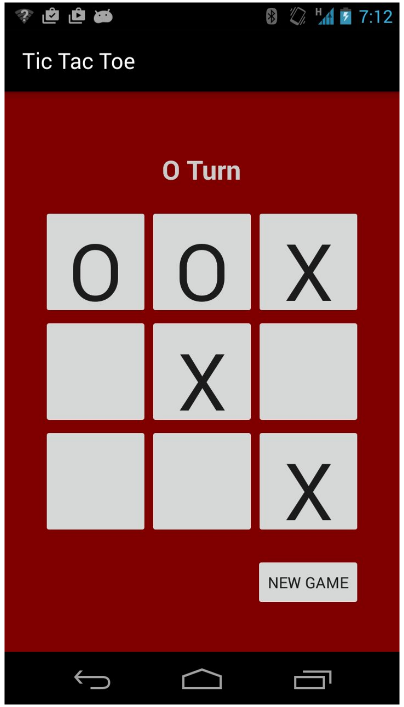
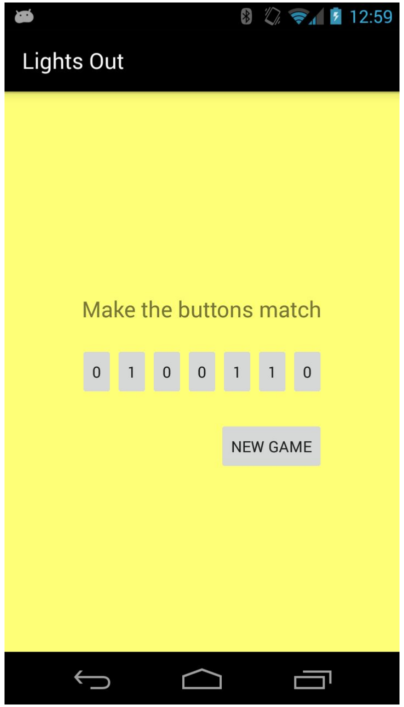
Buttons and the Model-View-Controller Pattern



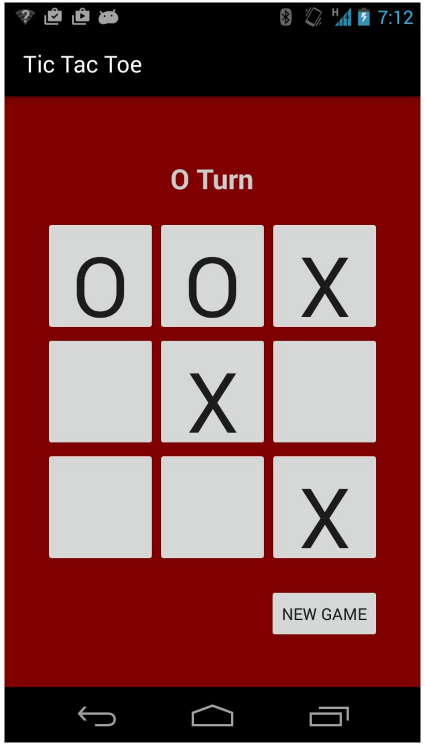




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

By the end of this unit you should be able to...

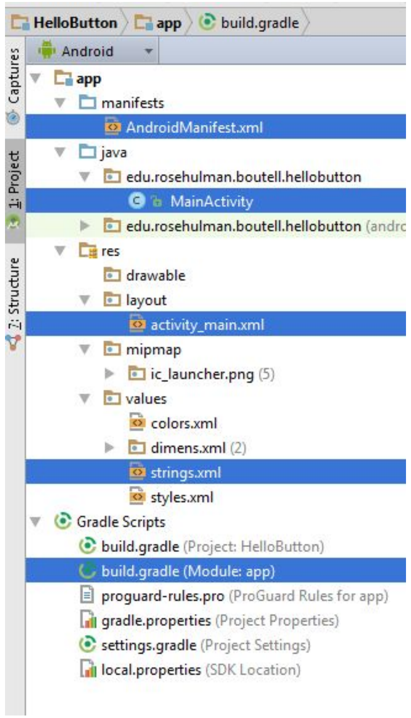
- ❑ Build apps that use buttons
- ❑ Build simple UIs to specification
  - ❑ Linear, relative, and table layouts
- ❑ Declare and use various resource values
  - ❑ Strings
  - ❑ Colors
  - ❑ Views
- ❑ Explain how Android supports the Model-View-Controller (MVC) pattern



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

New Android Application Project

In this lesson you will learn how to create a new Android project and to understand the various parts you are given



```
HelloButton > app > build.gradle
Android
▼ app
    ▼ manifests
          AndroidManifest.xml
    ▼ java
        ▼ edu.rosehulman.boutell.hellobutton
              MainActivity
        ▶ edu.rosehulman.boutell.hellobutton (andrc
    ▼ res
          drawable
        ▼ layout
              activity_main.xml
        ▼ mipmap
            ▶ ic_launcher.png (5)
        ▼ values
              colors.xml
            ▶ dimens.xml (2)
              strings.xml
              styles.xml
▼ Gradle Scripts
      build.gradle (Project: HelloButton)
      build.gradle (Module: app)
      proguard-rules.pro (ProGuard Rules for app)
      gradle.properties (Project Properties)
      settings.gradle (Project Settings)
      local.properties (SDK Location)
```

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Create a new Android Application Project

Android Studio > File > New … > New Project...

The defaults are pretty good, but follow these standards:

Use a descriptive app name: **HelloButton**

- for exams, will require your username, like **Exam1fFisher**

Company domain: use your name, like **fisher.rosehulman.edu**

Project location: anywhere you can find it.

- there is no concept of a *workspace*.

Default is good (Min SDK of 4.0.3 / API 15)

Empty Activity for now

Activity Name: MainActivity is often descriptive enough.

What are the key files?

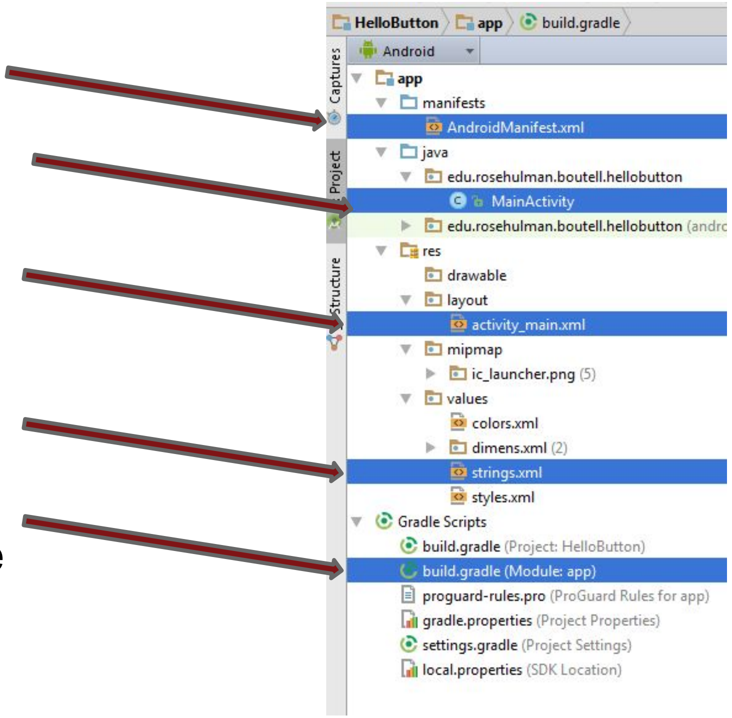
app permissions

model and controller Java code

view xml

all screen text

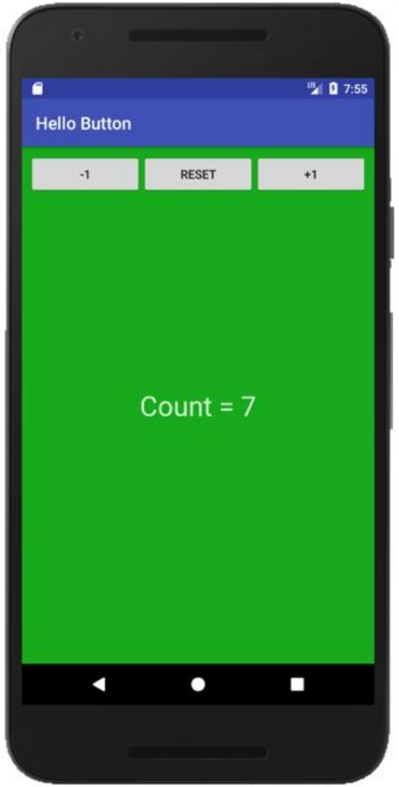
which external libraries to use



```
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▼ edu.rosehulman.boutell.hellobutton
        MainActivity
    ▶ edu.rosehulman.boutell.hellobutton (andrc
  ▼ res
      drawable
    ▼ layout
        activity_main.xml
    ▼ mipmap
      ▶ ic_launcher.png (5)
    ▼ values
        colors.xml
      ▶ dimens.xml (2)
        strings.xml
        styles.xml
▼ Gradle Scripts
    build.gradle (Project: HelloButton)
    build.gradle (Module: app)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

Hello Button View

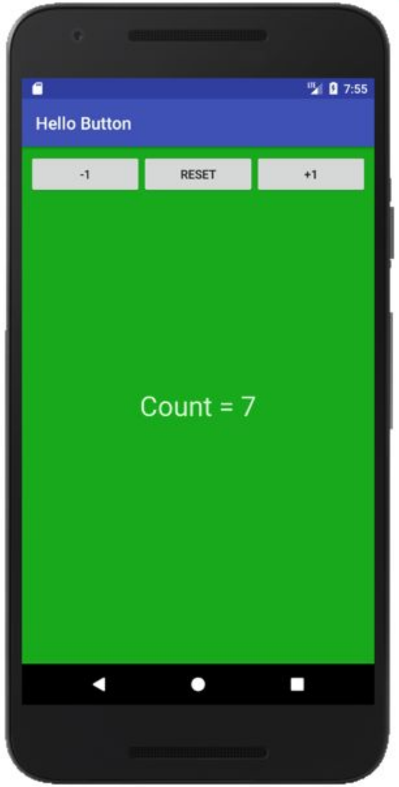
In this lesson you will learn how to implement a UI from a specification



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Implementing a UI from a specification

Start by planning resources: strings, colors



Plan by adding strings to strings.xml to make it easy to update later. Like translating to a new language!

Plan your resources if possible:

```
<resources>
  <string name="app_name">Hello Button</string>
  <string name="message_start">Count = 0</string>
  <string name="message_format">Count = %d</string>
  <string name="button_decrement">-1</string>
  <string name="button_reset">Reset</string>
  <string name="button_increment">+1</string>
</resources>
```

Note: There is also a thing called ‘plurals’ that make this more elegant, but more complex. Given that this is our first app, I won’t bother with ‘plurals’ today. Read more here: http://developer.android.com/guide/topics/resources/string-resource.html

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Create color resources: colors.xml

File > New > Other > Android xml file if it isn't there.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="colorPrimary">#3F51B5</color>
   <color name="colorPrimaryDark">#303F9F</color>
   <color name="colorAccent">#FF4081</color>
   <color name="background">#ff00aa00</color>
   <color name="text">#dfd</color>
</resources>
```

#<alpha><red><green><blue>

Colors given in RGB or ARGB format (A = alpha = transparency)
Background is bright green: (red, green, blue) = (0, 170, 0)
Text is very light green: (238, 255, 238)
Each value can be **1 or 2 hex digits, alpha is optional**

Implementing a UI from a specification

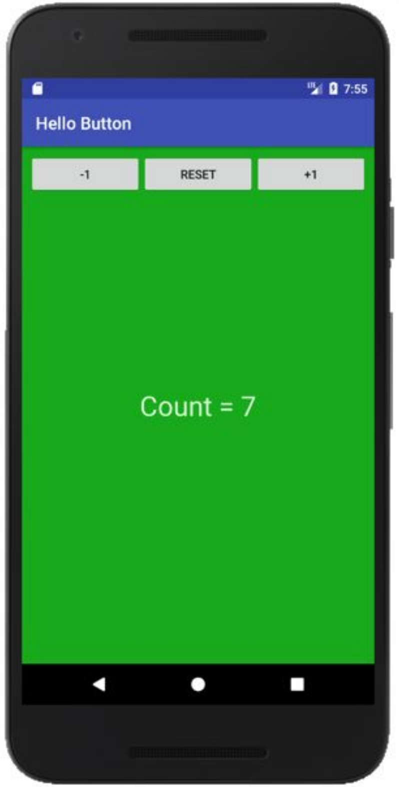
Observe:

- Text is centered in both directions, 32 sp, light green
- LinearLayout has a 8 dp margin on all sides
- Buttons equally fill the Linear Layout
- Green background

When elements are centered in screen or defined in relation to each other (aligned, above, below, etc), a **RelativeLayout** often works well. Even spacing in a row or column a **LinearLayout** often works well. Nesting layouts deeply is bad, this one is ok though. :)

Implement in layout > activity_main.xml now!

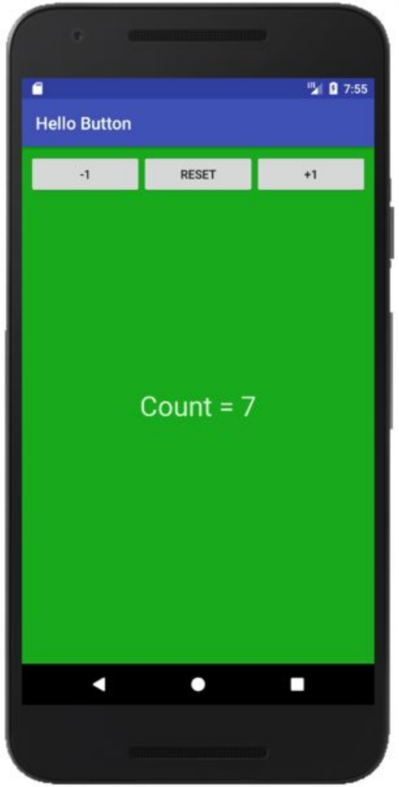
dp = density-independent pixels. Like px, but works with multiple resolutions. **Use for layouts.**
sp = scale-independent pixels. Like dp, but scaled by user's font size preference. **Use for text size.**
http://developer.android.com/guide/topics/resources/more-resources.html#Dimension



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

A Solution

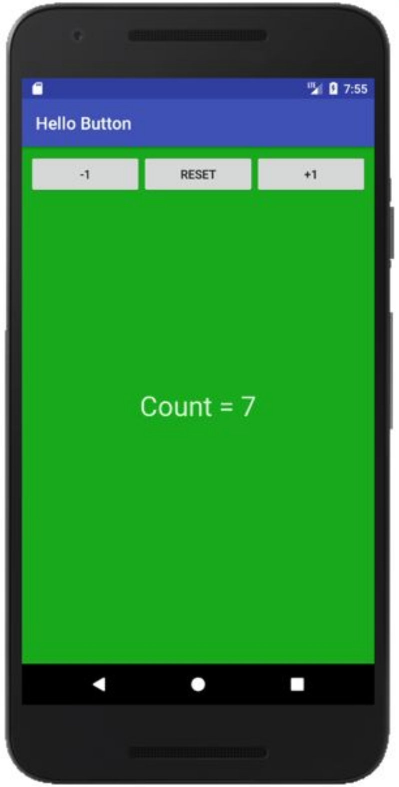
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"   android:layout_height="match_parent"
  android:background="@color/background"
  tools:context="edu.rosehulman.fisher.hellobutton.MainActivity">

  <LinearLayout
    android:layout_width="match_parent"      android:layout_height="wrap_content"
    android:layout_margin="8dp"      android:orientation="horizontal">
    <Button
      android:layout_width="0dp"        android:layout_height="wrap_content"
      android:layout_weight="1"        android:text="@string/button_decrement"  />
    <Button
      android:layout_width="0dp"        android:layout_height="wrap_content"
      android:layout_weight="1"        android:text="@string/button_reset"  />
    <Button
      android:layout_width="0dp"        android:layout_height="wrap_content"
      android:layout_weight="1"        android:text="@string/button_increment"  />
  </LinearLayout>

  <TextView
    android:id="@+id/message_text_view"      android:layout_width="wrap_content"
    android:layout_height="wrap_content"      android:text="@string/message_start"
    android:layout_centerHorizontal="true"     android:layout_centerVertical="true"
    android:textColor="@color/text"      android:textSize="32sp"/>

</RelativeLayout>
```



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Hello Button Controller

In this lesson you will learn how to refer to resources in code and to create the button logic



Making an instance variable (field) to track the state

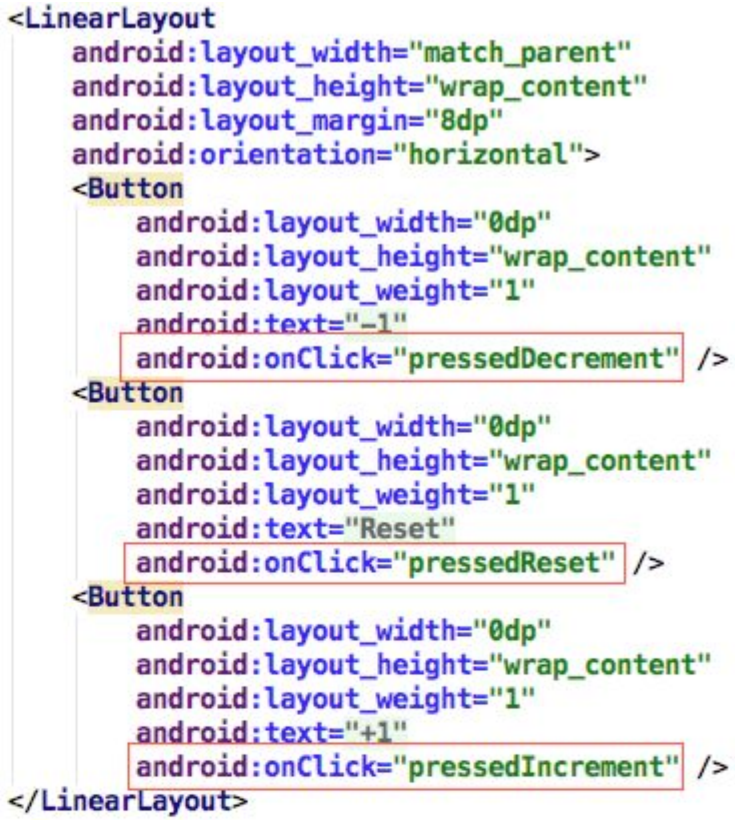
```
public class MainActivity extends AppCompatActivity {

    private int count = 0;

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

    }
}
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Capture references to Views using findViewById()

```
public class MainActivity extends AppCompatActivity {
        private int count = 0;
        private TextView textView;

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

                textView = (TextView) findViewById(R.id.message_text_view);
        }
}
```

Set a callback for button clicks



```xml
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:orientation="horizontal">
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="-1"
        android:onClick="pressedDecrement" />
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Reset"
        android:onClick="pressedReset" />
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="+1"
        android:onClick="pressedIncrement" />
</LinearLayout>
```

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Set a callback for button clicks

```
public class MainActivity extends AppCompatActivity {

 private int count = 0;
 private TextView textView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   textView = (TextView) findViewById(R.id.message_text_view);
 }

 public void pressedIncrement(View view) {
   count++;
   updateView();
 }

 public void pressedDecrement(View view) {
   count--;
   updateView();
 }

 public void pressedReset(View view) {
   count = 0;
   updateView();
 }
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Set a callback for button clicks

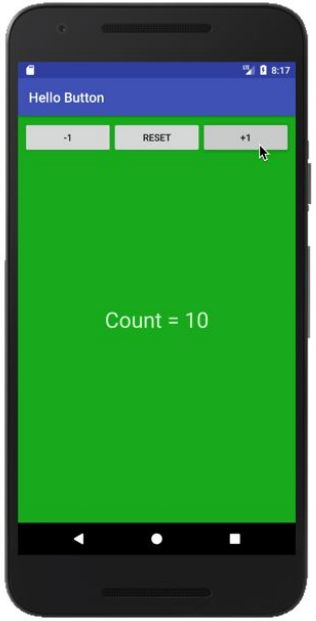
Demonstrating log (very useful) and UI updates

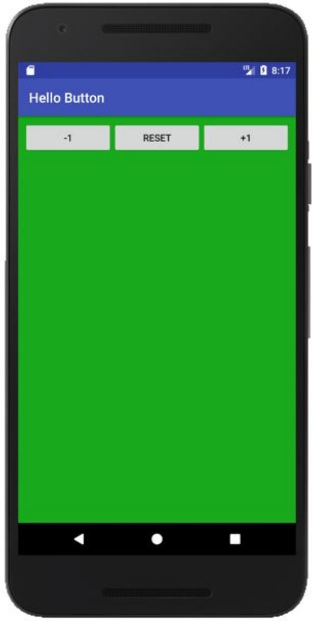
```
public void pressedIncrement(View view) {
  count++;
  updateView();
}

public void pressedDecrement(View view) {
  count--;
  updateView();
}

public void pressedReset(View view) {
  count = 0;
  updateView();
}

public void updateView() {
  Log.d("HelloButton", "Count updated to " + count);
  textView.setText(getString(R.string.message_format, count));
}
```

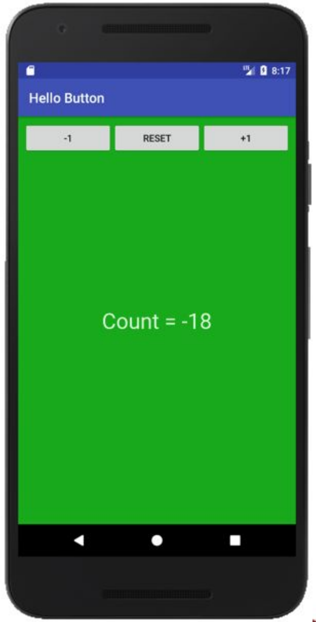
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Challenge

See if you can make the text **visible** when count <= 10, but **invisible when the count > 10**
Hint: .setVisibility(View.*INVISIBLE*)



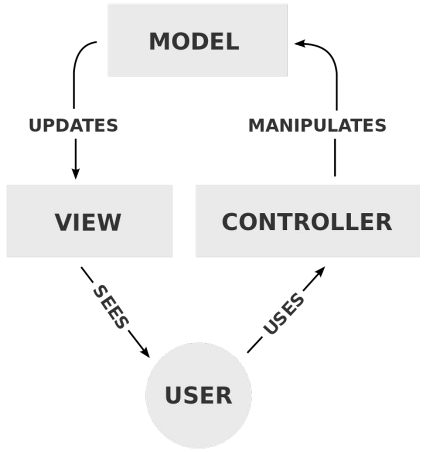




Note, it should disappear >10, but come back if count is ever <=10

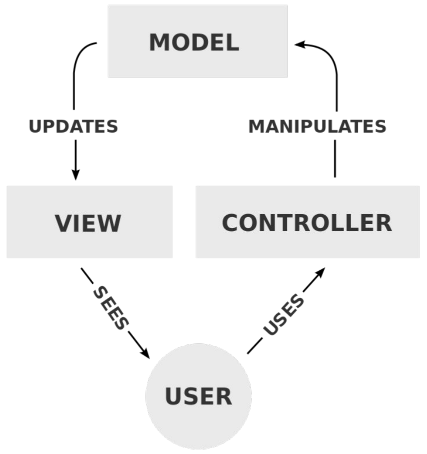
Model View Controller (MVC) and Tic Tac Toe Model

In this lesson you will learn how Android helps you follow the MVC paradigm



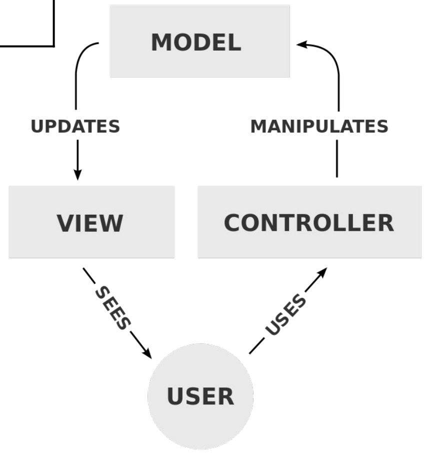
The MVC design pattern gives clean separation of the parts of interactive programs

Model	Our data
View	The display
Controller	User input



MVC with Android and Tic Tac Toe

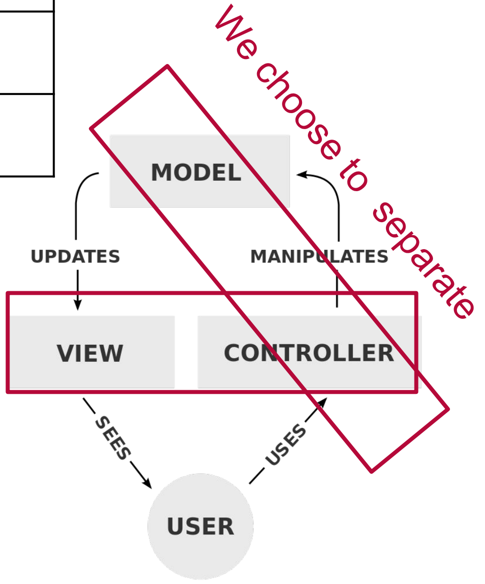
MVC	Android	TicTacToe
Model	Java classes	TicTacToeGame (array of marks, win logic)
View	activity_main.xml	Table layout
Controller	MainActivity.java	Multiple button listeners



MVC with Android and Tic Tac Toe

MVC	Android	TicTacToe
Model	Java classes	TicTacToeGame (array of marks, win logic)
View	activity_main.xml	Table layout
Controller	MainActivity.java	Multiple button listeners

Android separates



Download the provided files (model and icon)

Download Tic-Tac-Toe files

Unzip them and have them ready

Create a new Android Application Project

Project Name: Tic Tac Toe

Min SDK: 4.03

Company domain: *username*.rosehulman.edu

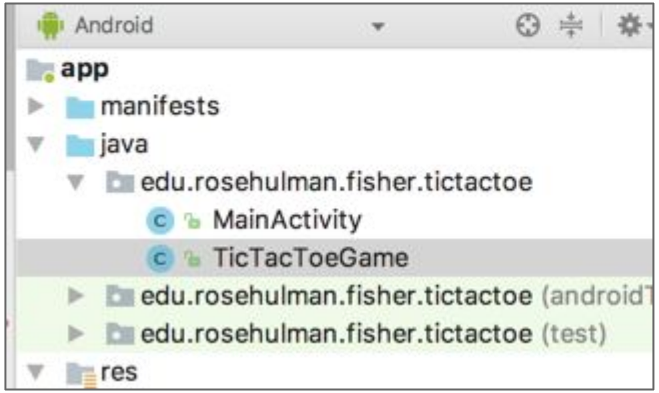
Empty Activity

Add the model file

Drag and drop the TicTacToe model file into the same package (folder) as your MainActivity

Take a minute to see what you have:

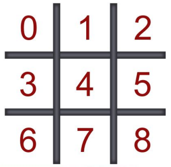
- storage for the board
- make a move
- check for win



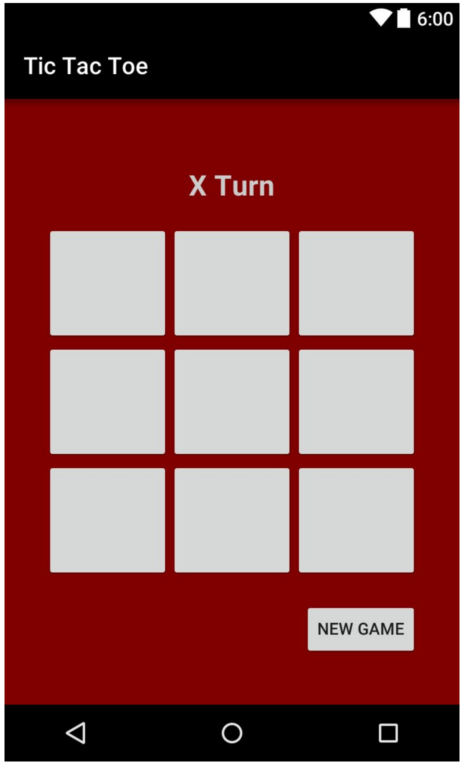
```
app
  manifests
  java
    edu.rosehulman.fisher.tictactoe
      MainActivity
      TicTacToeGame
    edu.rosehulman.fisher.tictactoe (android
    edu.rosehulman.fisher.tictactoe (test)
  res
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

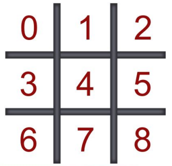
Tic Tac Toe View



In this lesson you will learn how to create tables and RelativeLayouts



Implementing a UI from a specification



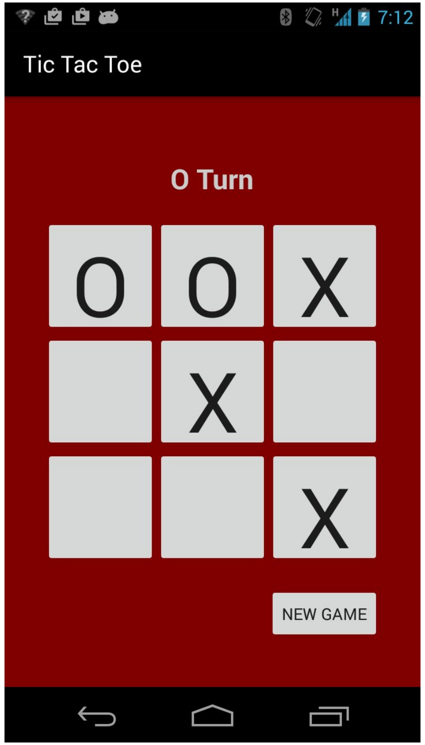
Observe:

- Fixed 3x3 table, centered in both directions, 20 dp (pixel) margin all around
- Game state **above** and **center**, 24 pt bold
- Game state is one of {X Turn, Y Turn, X Wins, Y Wins, Tie Game}
- New game button **below** and **right-aligned** with table
- Red background and gray text
- Buttons are 100dp high with 72sp text

When elements are defined in relation to each other (aligned, above, below, etc), use a **RelativeLayout**

Start in XML with the one whose position you care most about

You could figure this out. Let's try now.



strings.xml

```xml
<resources>
  <string name="app_name">Tic-Tac-Toe</string>
  <string name="new_game">New Game</string>
  <string name="x_turn">X\'s Turn</string>
  <string name="o_turn">O\'s Turn</string>
  <string name="x_win">X Wins!</string>
  <string name="o_win">O Wins!</string>
  <string name="tie_game">Tie Game</string>
</resources>
```

New > Android XML file: colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#3F51B5</color>
  <color name="colorPrimaryDark">#303F9F</color>
  <color name="colorAccent">#FF4081</color>

  <color name="textColor">#ccc</color>
  <color name="background">#800000</color>

</resources>
```

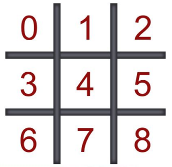
🟥
⬜

#<alpha><red><green><blue>

I changed the default colors to black and gray and added a background color (official Rose-Hulman red). We'll use the accent color for text.

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Where we are going: this incomplete xml



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
         xmlns:tools="http://schemas.android.com/tools"
         android:background="@color/background" >
  <TableLayout>
            <TableRow>
                <Button android:id="@+id/ button0"/>
                <Button android:id="@+id/ button1"/>
                <Button android:id="@+id/ button2"/>
            </TableRow>
            <TableRow>
                <Button android:id="@+id/ button3"/>
                <Button android:id="@+id/ button4"/>
                <Button android:id="@+id/ button5"/>
            </TableRow>
            <TableRow>
                <Button android:id="@+id/ button6"/>
                <Button android:id="@+id/ button7"/>
                <Button android:id="@+id/ button8"/>
            </TableRow>
  </TableLayout>
  <TextView android:id="@+id/ game_state_text_view"/>
  <Button android:id="@+id/ new_game_button" android:text="@string/new_game" />
</RelativeLayout>
```

Use a TableLayout for a matrix of buttons

If marked as stretchable, it can expand in width to fit any extra space. The total width of the table is defined by its parent container.

You can stretch all columns by using the value "*".
http://developer.android.com/reference/android/widget/TableLayout.html

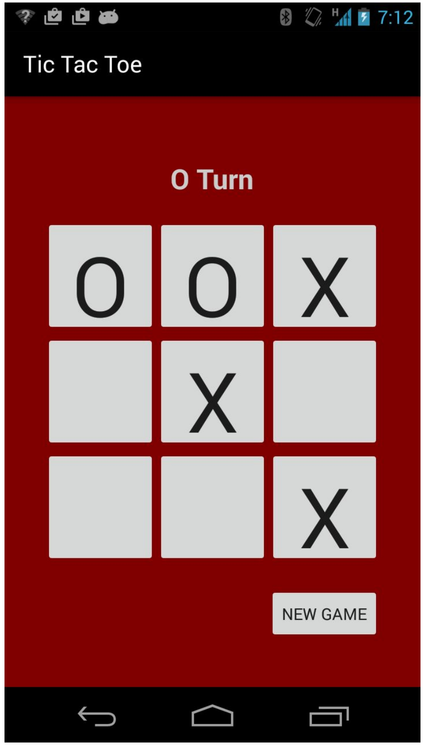
Reminder about buttons:
height = 100dp
width not needed due to stretchColumns
id’s are row col: button02 for top right

```
  <TableLayout
     android:id="@+id/table"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_centerHorizontal="true"
     android:layout_centerVertical="true"
     android:layout_margin="20dp"
     android:stretchColumns="*" >

  <TableRow>
  </TableRow>
  <TableRow>
  </TableRow>
  <TableRow>
  </TableRow>
</TableLayout>
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Tic Tac Toe Controller

In this lesson you will learn how to refer to your model and how to listen to multiple buttons



Controller: Each button affects the model and the view

Setup:

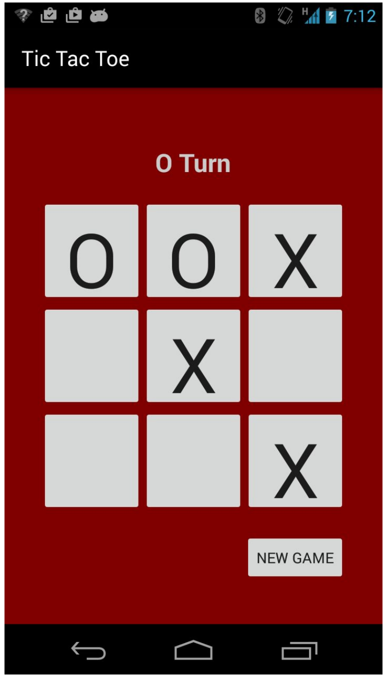
1. Create an instance of the model
2. Capture buttons, add listeners

Tic Tac Toe Buttons:

1. Tell the model that space was pressed
2. Change text on buttons and on game state

New game button:

1. Tell the game to reset
2. Change text on buttons and on game state



Capture the views

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private TicTacToeGame mGame = new TicTacToeGame(this);
    private Button[][] mTicTacToeButtons;
    private TextView mGameStateTextView;

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

        mGameStateTextView = (TextView) findViewById(R.id.message_text);

        Button newGameButton = (Button) findViewById(R.id.new_game_button);
        newGameButton.setOnClickListener(this);

        mTicTacToeButtons = new Button[TicTacToeGame.NUM_ROWS][TicTacToeGame.NUM_COLUMNS];
        for (int row = 0; row < TicTacToeGame.NUM_ROWS; row++) {
            for (int col = 0; col < TicTacToeGame.NUM_COLUMNS; col++) {
                int id = getResources().getIdentifier("button" + row + col, "id", getPackageName());
                mTicTacToeButtons[row][col] = (Button) findViewById(id);
                mTicTacToeButtons[row][col].setOnClickListener(this);
            }
        }
    }
```

Building an ID programmatically lets us avoid capturing 9 buttons without a loop

Set onClickListeners

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private TicTacToeGame mGame = new TicTacToeGame(this);
    private Button[][] mTicTacToeButtons;
    private TextView mGameStateTextView;

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

        mGameStateTextView = (TextView) findViewById(R.id.message_text);

        Button newGameButton = (Button) findViewById(R.id.new_game_button);
        newGameButton.setOnClickListener(this);

        mTicTacToeButtons = new Button[TicTacToeGame.NUM_ROWS][TicTacToeGame.NUM_COLUMNS];
        for (int row = 0; row < TicTacToeGame.NUM_ROWS; row++) {
            for (int col = 0; col < TicTacToeGame.NUM_COLUMNS; col++) {
                int id = getResources().getIdentifier("button" + row + col, "id", getPackageName());
                mTicTacToeButtons[row][col] = (Button) findViewById(id);
                mTicTacToeButtons[row][col].setOnClickListener(this);
            }
        }
    }
}
```

Alternate style: have the activity (this) be the listener. So onClick() will be part of MainActivity.

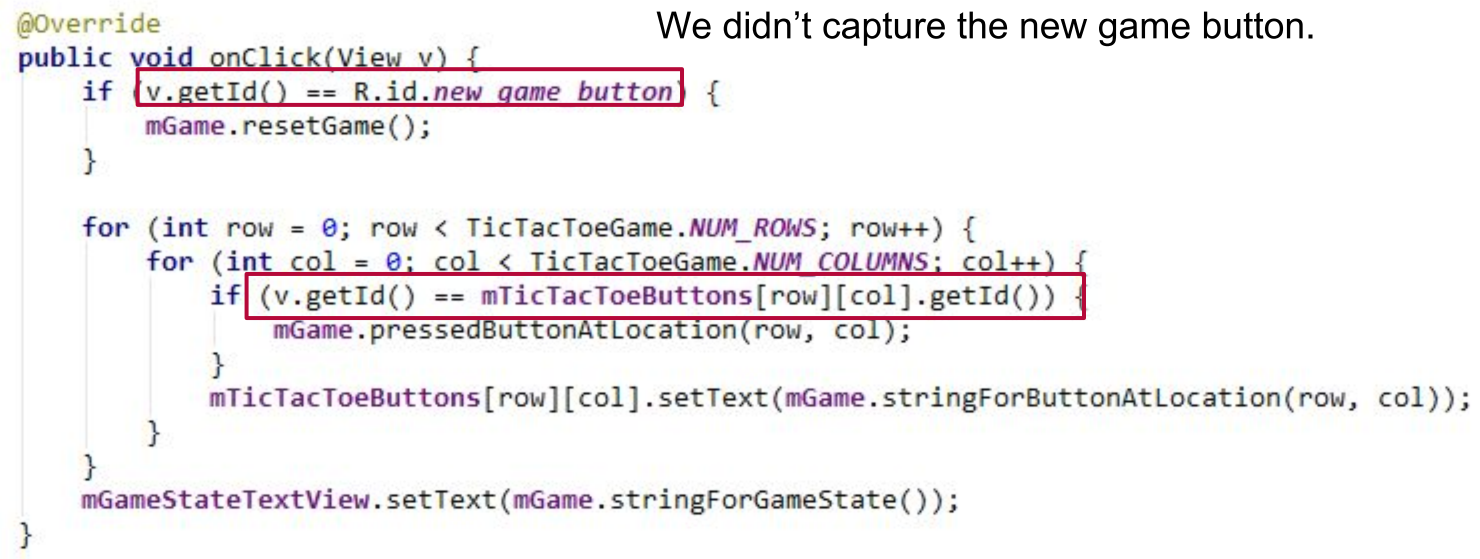
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

OnClick(): call game's pressedButtonAtLocation() method

Can we do better?

```
@Override
public void onClick(View v) {
      switch (v.getId()) {
      case R.id.newGame:
         mGame.resetGame();
         break;
      case R.id.button00:
         mGame.pressedButtonAtLocation(0, 0);
         break;
      case R.id.button01:
         mGame.pressedButtonAtLocation(0, 1);
         break;
     //...
      case R.id.button22:
        mGame.pressedButtonAtLocation(2, 2);
        break;
      }
}
```

Detect which button is pressed using its ID

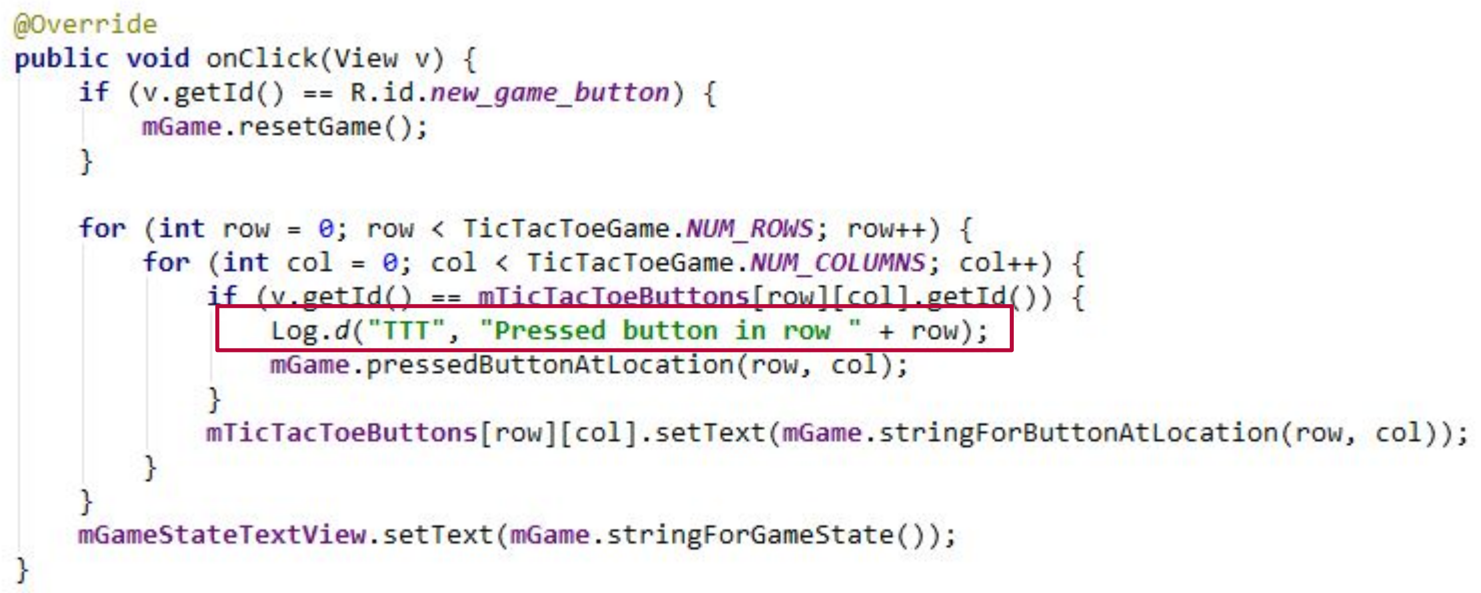


```
@Override
public void onClick(View v) {
    if (v.getId() == R.id.new_game_button) {
        mGame.resetGame();
    }

    for (int row = 0; row < TicTacToeGame.NUM_ROWS; row++) {
        for (int col = 0; col < TicTacToeGame.NUM_COLUMNS; col++) {
            if (v.getId() == mTicTacToeButtons[row][col].getId()) {
                mGame.pressedButtonAtLocation(row, col);
            }
            mTicTacToeButtons[row][col].setText(mGame.stringForButtonAtLocation(row, col));
        }
    }
    mGameStateTextView.setText(mGame.stringForGameState());
}
```

We didn't capture the new game button.

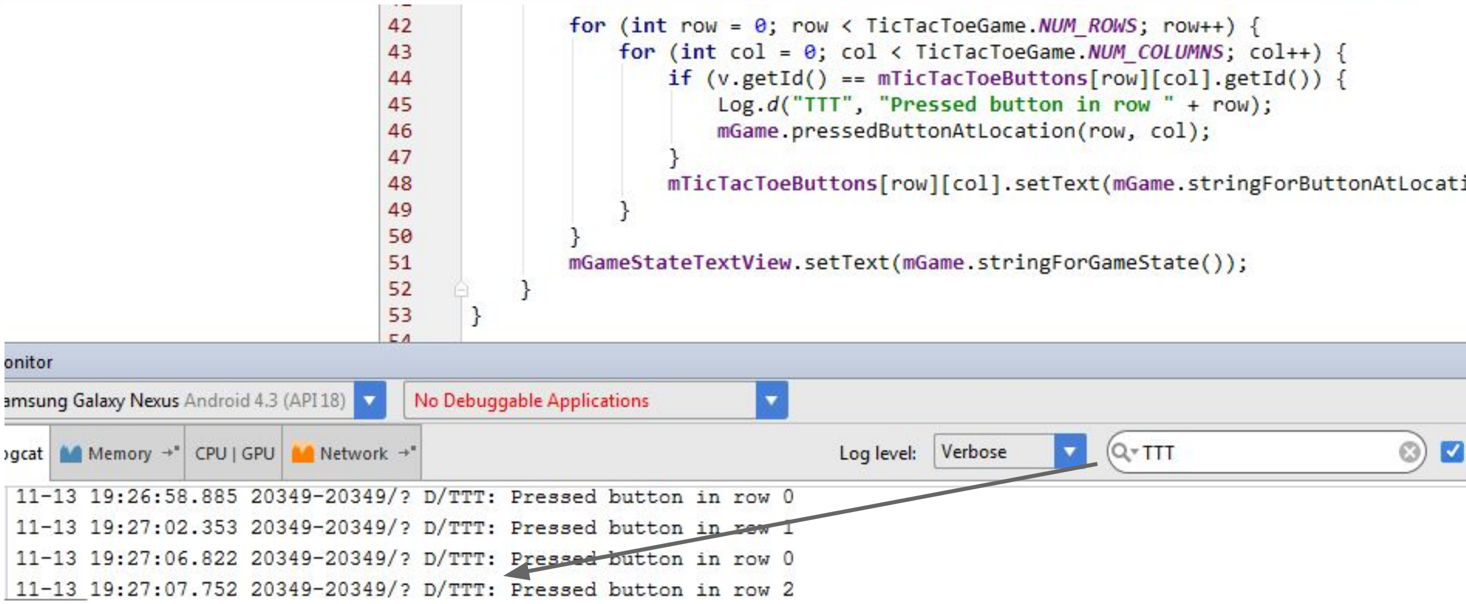
We captured these buttons in a 2d array.

Get test output by printing a Log message.



```
@Override
public void onClick(View v) {
    if (v.getId() == R.id.new_game_button) {
        mGame.resetGame();
    }

    for (int row = 0; row < TicTacToeGame.NUM_ROWS; row++) {
        for (int col = 0; col < TicTacToeGame.NUM_COLUMNS; col++) {
            if (v.getId() == mTicTacToeButtons[row][col].getId()) {
                Log.d("TTT", "Pressed button in row " + row);
                mGame.pressedButtonAtLocation(row, col);
            }
            mTicTacToeButtons[row][col].setText(mGame.stringForButtonAtLocation(row, col));
        }
    }
    mGameStateTextView.setText(mGame.stringForGameState());
}
```

Filter log messages by tag to make them easy to find



```
            for (int row = 0; row < TicTacToeGame.NUM_ROWS; row++) {
                for (int col = 0; col < TicTacToeGame.NUM_COLUMNS; col++) {
                    if (v.getId() == mTicTacToeButtons[row][col].getId()) {
                        Log.d("TTT", "Pressed button in row " + row);
                        mGame.pressedButtonAtLocation(row, col);
                    }
                    mTicTacToeButtons[row][col].setText(mGame.stringForButtonAtLocati
                }
            }
            mGameStateTextView.setText(mGame.stringForGameState());
        }
    }
```

onitor
amsung Galaxy Nexus Android 4.3 (API 18)
No Debuggable Applications
ogcat
Memory
CPU | GPU
Network
Log level:
Verbose
TTT

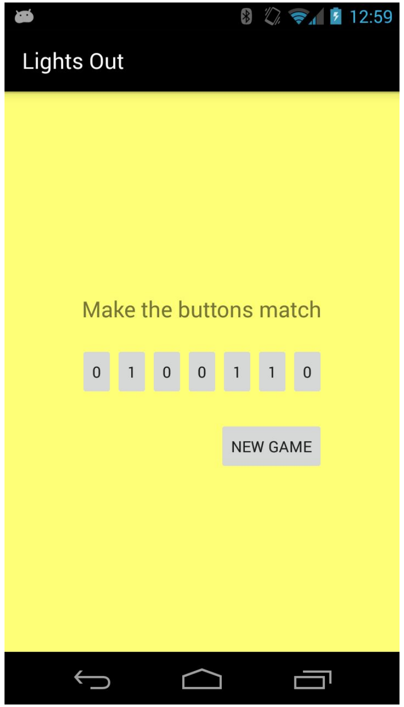
```
11-13 19:26:58.885 20349-20349/? D/TTT: Pressed button in row 0
11-13 19:27:02.353 20349-20349/? D/TTT: Pressed button in row 1
11-13 19:27:06.822 20349-20349/? D/TTT: Pressed button in row 0
11-13 19:27:07.752 20349-20349/? D/TTT: Pressed button in row 2
```

Lab: Lights Out

Your turn.

Win a game, reset, rotate.

See the schedule page for the link.

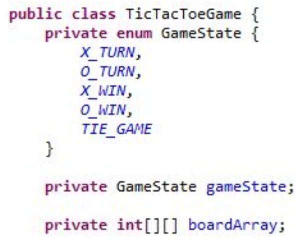


Summary: Simple interactive apps use resources, layouts, and listeners in code.

Upon completion of this unit, you will be able to...

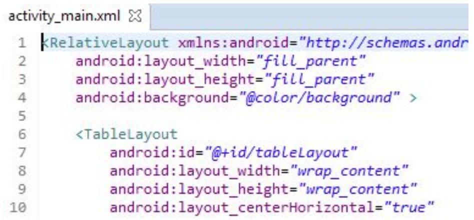
- ❑ Use values (strings.xml, colors.xml)
- ❑ Create a layout (drag and drop, properties view, raw xml)
- ❑ Reference views and values from code
- ❑ Create button listeners
- ❑ Use the MVC paradigm

Model



```
public class TicTacToeGame {
    private enum GameState {
        X_TURN,
        O_TURN,
        X_WIN,
        O_WIN,
        TIE_GAME
    }

    private GameState gameState;

    private int[][] boardArray;
```

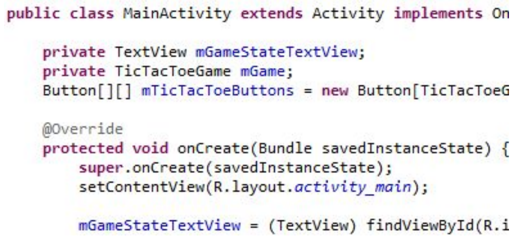
View



activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.andr
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/background" >

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
```

Controller



```
public class MainActivity extends Activity implements On

    private TextView mGameStateTextView;
    private TicTacToeGame mGame;
    Button[][] mTicTacToeButtons = new Button[TicTacToeG

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

        mGameStateTextView = (TextView) findViewById(R.i
```