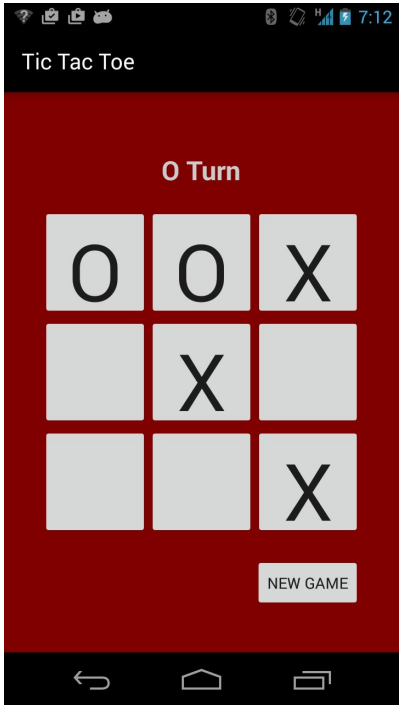
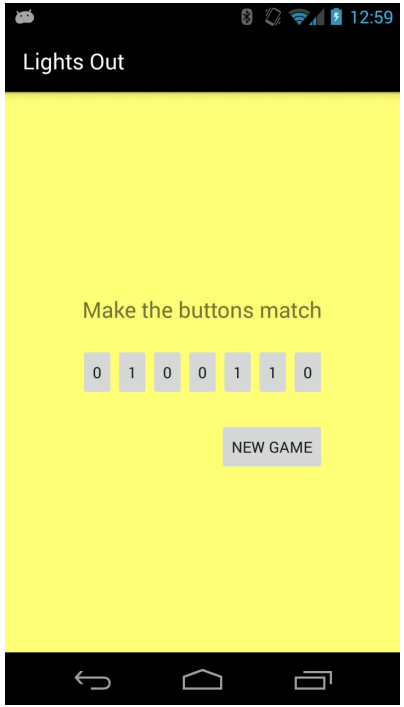
Buttons and the Model-View-Controller Pattern

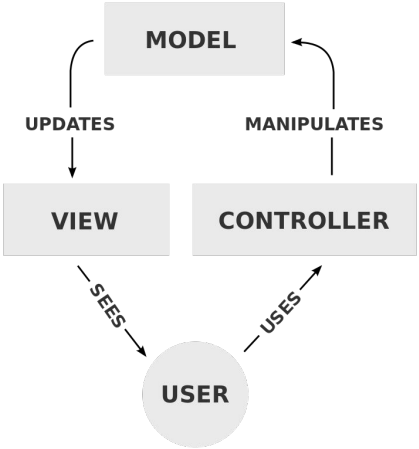




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

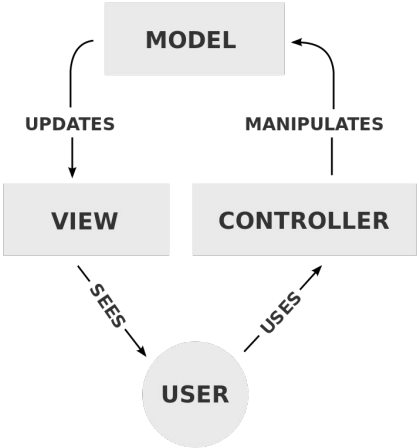
Model View Controller (MVC) and Tic Tac Toe Model

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

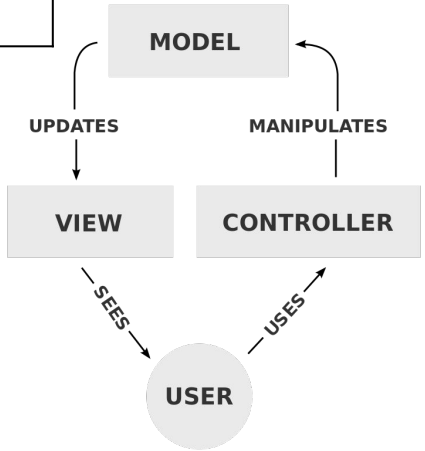
Model	Our data
View	The display
Controller	User input



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

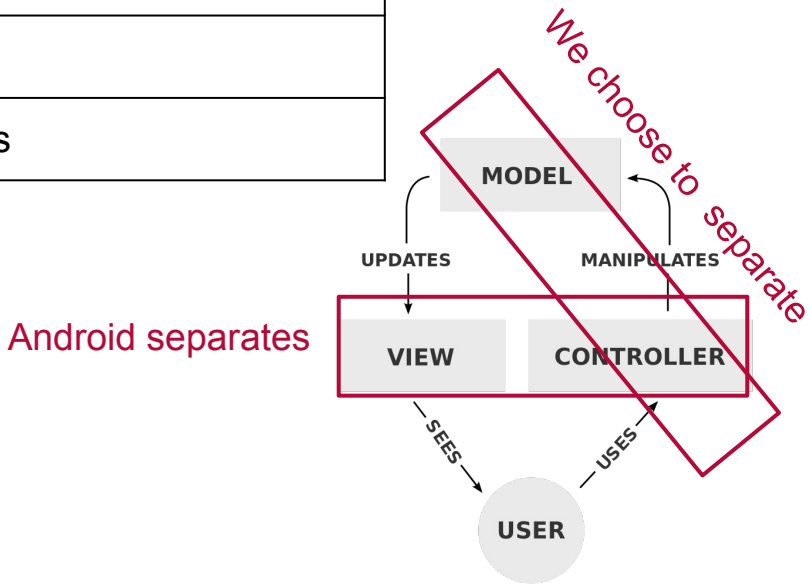
MVC with Android and Tic Tac Toe

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



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

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

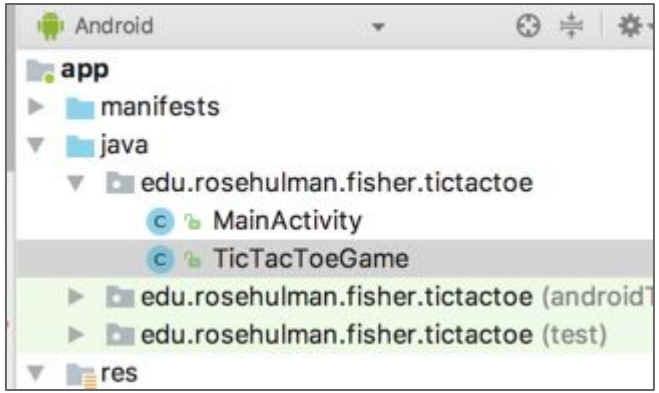
Empty Activity

Add the model file

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

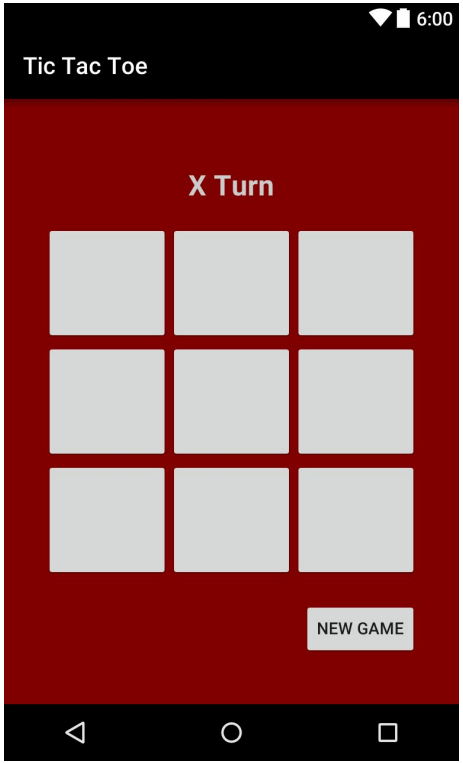
Take a minute to see what you have:

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



Tic Tac Toe View

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



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Implementing a UI from a specification

0	1	2
3	4	5
6	7	8

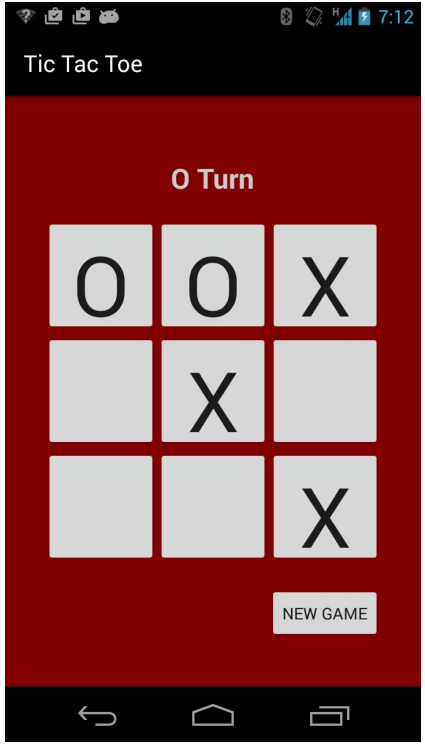
Observe:

- 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.



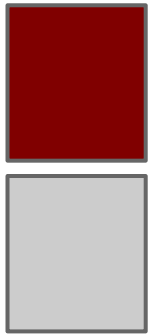
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

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

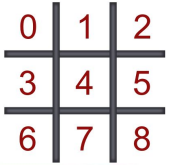
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#3F51B5</color>
  <color name="colorPrimaryDark">#303F9F</color>
  <color name="colorAccent">#FF4081</color>

  <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.

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

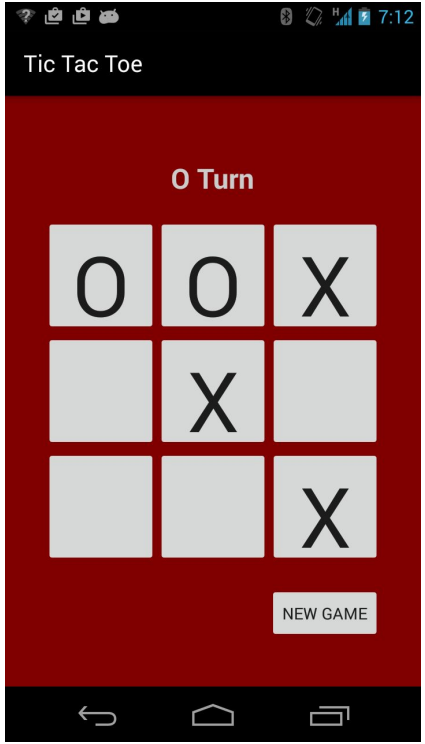
Reminder about buttons:
height = 100dp
width not needed due to stretchColumns
id's are row col: button02 for top right

```
<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



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Controller: Each button affects the model and the view

Setup:

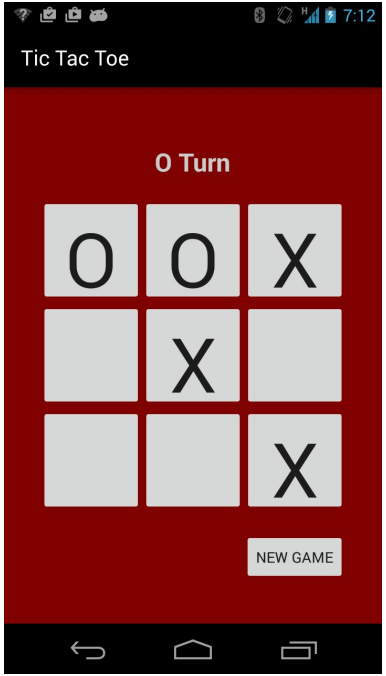
1. Create an instance of the model
2. Capture buttons, add listeners

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

```java
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

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Set onClickListeners

```java
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;
      }
}
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

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

```
42          for (int row = 0; row < TicTacToeGame.NUM_ROWS; row++) {
43              for (int col = 0; col < TicTacToeGame.NUM_COLUMNS; col++) {
44                  if (v.getId() == mTicTacToeButtons[row][col].getId()) {
45                      Log.d("TTT", "Pressed button in row " + row);
46                      mGame.pressedButtonAtLocation(row, col);
47                  }
48                  mTicTacToeButtons[row][col].setText(mGame.stringForButtonAtLocati
49              }
50          }
51          mGameStateTextView.setText(mGame.stringForGameState());
52      }
53  }
```

onitor

amsung Galaxy Nexus Android 4.3 (API 18) | No Debuggable Applications

ogcat | Memory | CPU | GPU | Network | Log level: Verbose | TTT

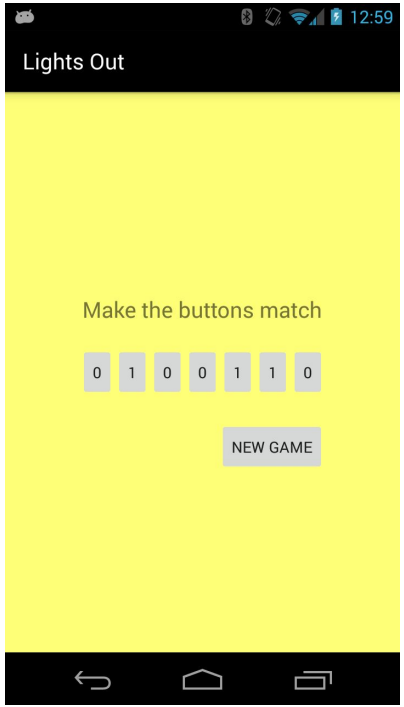
```
11-13 19:26:58.885 20349-20349/? D/TTT: Pressed button in row 0
11-13 19:27:02.353 20349-20349/? D/TTT: Pressed button in row 1
11-13 19:27:06.822 20349-20349/? D/TTT: Pressed button in row 0
11-13 19:27:07.752 20349-20349/? D/TTT: Pressed button in row 2
```

Lab: Lights Out

Your turn.

Win a game, reset, rotate.

See the schedule page for the link.



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

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

```
1  <RelativeLayout xmlns:android="http://schemas.andr
2      android:layout_width="fill_parent"
3      android:layout_height="fill_parent"
4      android:background="@color/background" >
5
6      <TableLayout
7          android:id="@+id/tableLayout"
8          android:layout_width="wrap_content"
9          android:layout_height="wrap_content"
10         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
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY