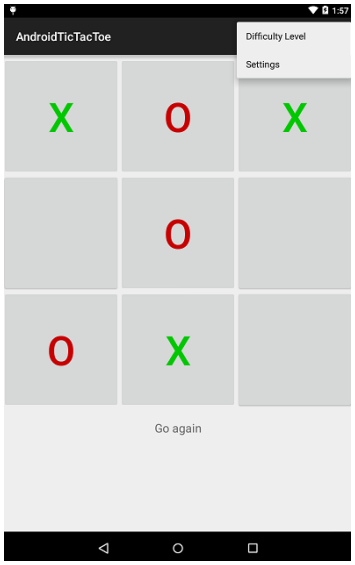
Android Application Programming

Tutorial 3: Menus and Dialog Boxes

Introduction

In the previous tutorial, you created a simple tic-tac-toe game for the Android. The goal of this tutorial is to improve your game by creating a more sofisticated menu that uses a few dialog boxes. In this tutorial you will create an options menu (as shown on the right) with three options:



1. **New Game** – to start a new game
2. **Difficulty** – to set the AI difficulty level to Easy, Harder, or Expert
3. **Settings** – to be done in a later tutorial

You should familiarize yourself with the official Menu Design Guidelines when designing your own menus. Android also supports *context menus*, floating lists of menu items that appear when holding your finger down on the screen (a long press), but they will not be covered in this tutorial.

Changes to the Game Logic

We are going to create a menu option in this tutorial to change the difficulty level of the game. Right now the game only has one difficulty level which we'll call "Expert". You will need to modify the `TicTacToeGame` class so the difficulty level can be set. When set to the "Easy" level, the computer will always just make random moves. When set to the "Harder" level, the computer will make a winning move if possible, otherwise it will make a random move.

1. Open **TicTacToeGame.java**, and add constants for the difficulty level and a variable for keeping track of the current difficulty level setting:

```
public class TicTacToeGame {

    // The computer's difficulty levels
    public static final int EASY = 0;
    public static final int HARDER = 1;
    public static final int EXPERT = 2;

    // Current difficulty level
    private int mDifficultyLevel;
```

2. Create getters and setters for the difficulty level:

```
public int getDifficultyLevel() {
   // to do
}

public void setDifficultyLevel(int difficultyLevel) {
   // to do
}
```

3. Finally, modify the `getComputerMove()` method to call the appropriate method depending on the difficulty level. **It's left to you** to implement `getRandomMove()`, `getWinningMove()`, and `getBlockingMove()` based on the pre-existing code. Note that the methods might need to *temporarily* modify the board array, but they should leave the array in the same state it was in before the methods were called.

```
public int getComputerMove() {

   int move = -1;

   if (mDifficultyLevel == EASY)
      move = getRandomMove();
   else if (mDifficultyLevel == HARDER)
   {
      move = getWinningMove();
      if (move == -1)
         move = getRandomMove();
   }
   else if (mDifficultyLevel == EXPERT)
   {

      // Try to win, but if that's not possible, block.
      // If that's not possible, move anywhere.
      move = getWinningMove();
      if (move == -1)
         move = getBlockingMove();
      if (move == -1)
         move = getRandomMove();
   }

   return move;
}
```

Create the Menu (Action Bar) in XML

Now we will add a menu to the Activity that will be used to set the TicTacToeGame's difficulty level. We'll have three options in our menu. You will need to add menu option for difficulty level to the current menu xml file shown below.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" tools:context=".TicTacToe">
    <item android:id="@+id/action_new_game" android:title="@string/action_new_game"
        android:orderInCategory="1" app:showAsAction="ifRoom" />

    <item android:id="@+id/action_settings" android:title="@string/action_settings"
                android:orderInCategory="100" app:showAsAction="never" />
</menu>
```

You need to have a menu option item for:

- New Game
- Difficulty
- Settings (we will use this later)

Display the Menu and Respond to Menu Selections

Now you need to add code to your program to create the menu and respond to menu items selections:

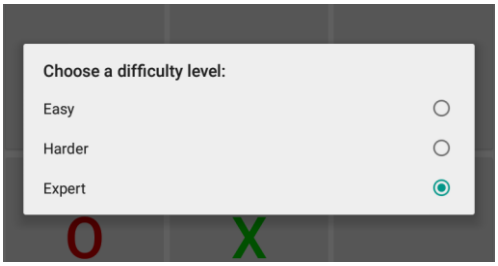
1. You need to modify the `onOptionsItemSelected()` method which is called when a menu option is selected. You can tell which menu item was selected by examining the menu item's ID which was given in the `menu xml` file.

   If the **`New Game`** option is selected, your program should call your `startNewGame()` method (already done).
   If the **`Difficulty`** option is selected, implement the program code below;

```
DialogFragment newFragment = new DifficultyDialog();
newFragment.show(getSupportFragmentManager(), "dialog");
```

The menu you just added will display a dialog box that correspond to some constants which we'll define next.

It will look something like what you see on the right.



2. You will next need to create a new class in your project. This needs to be named **DifficultyDialog.java** because you referred to it above in step 1.

Add the following program code into your new class.

```
public class DifficultyDialog extends DialogFragment
{
    private TicTacToeGame mGame;

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        Dialog dialog = null;
        mGame = ((TicTacToe)getActivity()).getGame();
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setTitle(R.string.difficulty_choose);

        final CharSequence[] levels = {
                getResources().getString(R.string.difficulty_easy),
                getResources().getString(R.string.difficulty_harder),
                getResources().getString(R.string.difficulty_expert)};

        // TODO: Set selected, an integer (0 to n-1), for the Difficulty dialog.

        // selected is the radio button that is selected when the Dialog Box appears.

        builder.setSingleChoiceItems(levels, selected,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int item) {
                        dialog.dismiss();   // Close dialog

                        // TODO: Set the diff level of mGame based on which item was selected.

                }
            });
        dialog = builder.create();

        return dialog;
    }
```

We will make use of the `AlertDialog` class which is useful for creating simple dialog boxes with 1-3 buttons.

In the code below, there are two places marked "TODO" where the code is incomplete. **You will need** to set the `selected` variable so the `setSingleChoiceItems()` method will properly show radio buttons labeled Easy, Harder, and Expert as shown on the right. The currently selected difficulty level should initially be selected.

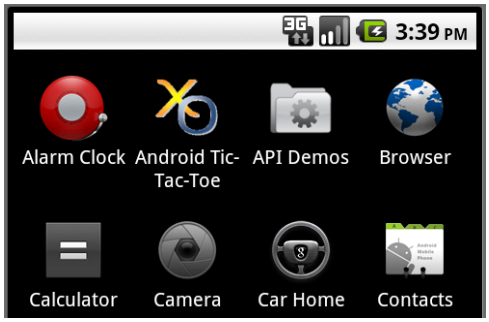
When one of the radio buttons is selected, the dialog will close, and the `item` parameter of `onClick` will indicate which radio button was selected. **You will need to write the code** to set the game's internal difficulty level. You will also need to add the appropriate string constants to **strings.xml**.

Hint #1: you will need to import **android.support.v4.app.DialogFragment**
Hint #2: each ToDo only takes one line of code.
Hint #3: you will need to write a getter method in TicTacToe.java to return a reference to **mGame**. You need the reference to **mGame** so that you can access the current difficulty level (to pre-populate the dialog) and so that you can change the difficulty level after the user clicks on the button of their choice.

Run your app and verify that the menu options work as expected.
Documentation for learning about different ways you can use Dialog Boxes is here:
http://developer.android.com/guide/topics/ui/dialogs.html

Extra Challenge

There are two extra challenges:

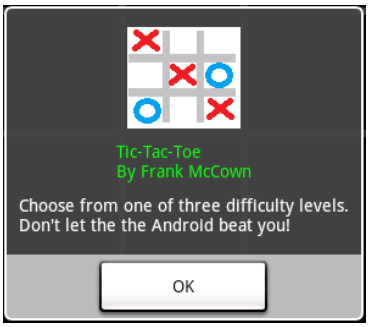


1. Create your own custom icon for your application. Name it `icon.png`, and place it in your `res/drawable` folder. Open your `AndroidManifest.xml` file, and alter `android:icon` to point to your new `icon.png`:

```
<application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
```

Now run your app. Press the Home key and press the App Drawer button on the screen to list all the apps installed. This is where you will see the icon you have created for your app. Mine is shown above.

2. Create a menu option that displays an About dialog box, similar the one shown on the right, which identifies you as the programmer.



Do a little research on the Android site on the previous page to figure out how to do a Dialog Box that fits your needs. It does not have to have an image as part of the Dialog but it adds a nice touch.