Assignment 3 : Instructions and Hints

May 2018

1 How do I start?

You can first start by looking into the tutorials provided in the Android Developers Website. Try creating a simple application at first, so that you get familiar with Android and Android studio, before delving into Assignment 3

2 Hints for Assignment 3

Once you gain knowledge of developing a simple Android application, you can make use of the following hints to start working on Assignment 3

2.1 How do I interact with the user?

The Activity class takes care of creating a window for you in which you can place your UI with `setContentView(View)`. More information regarding Activity can be found on this site. The entire lifecycle of an activity can be defined by implementing the following Activity methods:

```java

public class Activity extends ApplicationContext {
  protected void onCreate(Bundle savedInstanceState); //Invoked when the activity is first
     created
  protected void onStart(); //Invoked when the activity is becoming visible to the user
  protected void onRestart(); //Invoked after your activity has been stopped, prior to it being
     started again
  protected void onResume(); //Invoked when the activity will start interacting with the user. It
      is always followed by onPause()
  protected void onPause(); //Called when the system is about to start resuming a previous
     activity
  protected void onStop(); //Invoked when the activity is no longer visible to the user, because
     another activity has been resumed and is covering this one.
  protected void onDestroy(); //Called  before your activity is destroyed
}
```

Listing 1: Activity methods

All of these are the methods that you can override to do appropriate work when the activity changes state. Your Activity should implement `onCreate(Bundle)` to do the initial setup. You should always call up to your superclass when implementing these methods.

A Sample Activity class can be implemented as follows:

```java
public class MainActivity extends Activity {

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

  @Override
  protected void onPause() {
    super.onPause();
  }

  ...
  ...
  ...
  //Implement other functions that need to be called from the activity here.
}
```

Listing 2: Sample Activity

2.2 How do I draw something on the screen?

Now that you are aware of how to create a window for your application, you will need a surface to draw the candies. Android provides a class that allows you to render 2D bitmaps. This class is `SurfaceView`. You will want to extend this class to support the game functionality. You can read more about this class at
`https://developer.android.com/reference/android/view/SurfaceView.html`.
You might do this by declaring the `BoardView` class extends the `BoardView` class with the following declaration:
`public class BoardView extends SurfaceView {`.
The `onDraw` method is where all the action happens. You will want to override this method in your class, and use the method to draw the current snapshot of the screen.

Android provides great support for bitmaps. You can create a Bitmap object by using the following syntax:

```java
Bitmap mybitmap=BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher);
```

Listing 3: Bitmap Creation

This creates a bitmap of the ic_launcher.png file in res/drawable of the project. You can of course use other graphics by changing the names. The `drawable-ldpi`, `drawable-mdpi`, and `drawable-hdpi` directories store versions for devices with low, medium, and high resolution screens.

To draw a bitmap on the canvas, we override the `onDraw` method. For example:

```java
protected void onDraw(Canvas c) {
  c.drawColor(Color.BLACK);  // Set the background to black
  Rect dst=new Rect();
  dst.set(10, 30, 20, 40);  //Set window to place image from (10,30) to (20,40)
  c.drawBitmap(mybitmap, null, dst, null);  //Draw the bitmap
}
```

Listing 4: Draw bitmap on the Canvas

You can get the screen height and width by calling `getHeight` and `getWidth` in your program.

2.3 How do I detect screen touch events?

You have to first call `setFocusable(true);` in the initializer of your `BoardView` object. This tells Android to forward UI input events to your view.

Android will then call the `onTouchEvent` method when a touch event occurs. See Responding to Touch Events for more details.

To intercept these events, just override that method in your `BoardView` class.
Some relevant events for Candy Crush might be the following :

- `MotionEvent.ACTION_DOWN`

- `MotionEvent.ACTION_UP`
- `MotionEvent.ACTION_MOVE`

Feel free to use the ones most relevant to your implementation. You can get the coordinates by calling `getX` and `getY`. (Also, think about how you can capture the left and right screen swipe events.)

```java
package candycrush;

import ...;

// This is the "game engine''.
public class BoardView extends SurfaceView implements SurfaceHolder.Callback {
  public BoardView(Context context) {
    super(context);
    // Notify the SurfaceHolder that you'd like to receive SurfaceHolder callbacks.
    getHolder().addCallback(this);
    setFocusable(true); //Very important
    // Initialize game state variables and the game board variables.
    // DON'T RENDER THE GAME YET.
    ...
  }

  @Override
  public void surfaceCreated(SurfaceHolder holder) {
    // Construct game initial state
    ...
    //Create the Candy board
    //Initialize the board with random candies
  }

  @Override
  public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    // Called in response to changes in the surface.
    // Write code here to do the necessary tasks such as adding new random candies from the top
    of the board when the dimensions of the board change on eliminating the matched candies
  }

  @Override
  public void surfaceDestroyed(SurfaceHolder holder) {
    //This is called immediately before a surface is being destroyed.
    //Write code here that needs to be executed just before the surface is destroyed.
  }

  @Override
  public boolean onTouchEvent(MotionEvent e) {
    // Update game state in response to events:
    //  touch-down, touch-up, and touch-move.
    // Perform operations to check if the touch event is a valid move
    // If the move is valid, take the necessary actions such as removing the matched candies and
    moving the candies above the eliminated row/ column to their appropriate positions.
  }
}
```

Listing 5: SurfaceView Skeleton Code