

# Skin Guide for Developers

## Basic Preparation

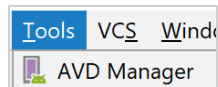
Before you start, you need to be a little bit familiar with these:

- Android developing IDE (Android Studio suggested)
- XML syntax for layout

But if you are not familiar with these, just check out some tutorial videos online, such as “android first app tutorial”, and you will be good to start. Very simple.

## Skin Making Progress

1. Start a new Android app project with minimum SDK API 19 in your Android developing IDE. (Package name shouldn't be too long)
2. You need to prepare to write down three names for skin controller to use later:
  - Layout name (i.e. `activity_main.xml`)
  - APK file name (i.e. `app-debug.apk`)
  - Package name (i.e. `cognitiveai.skin1`)
3. In AVD Manager (Android Virtual Device Manager)



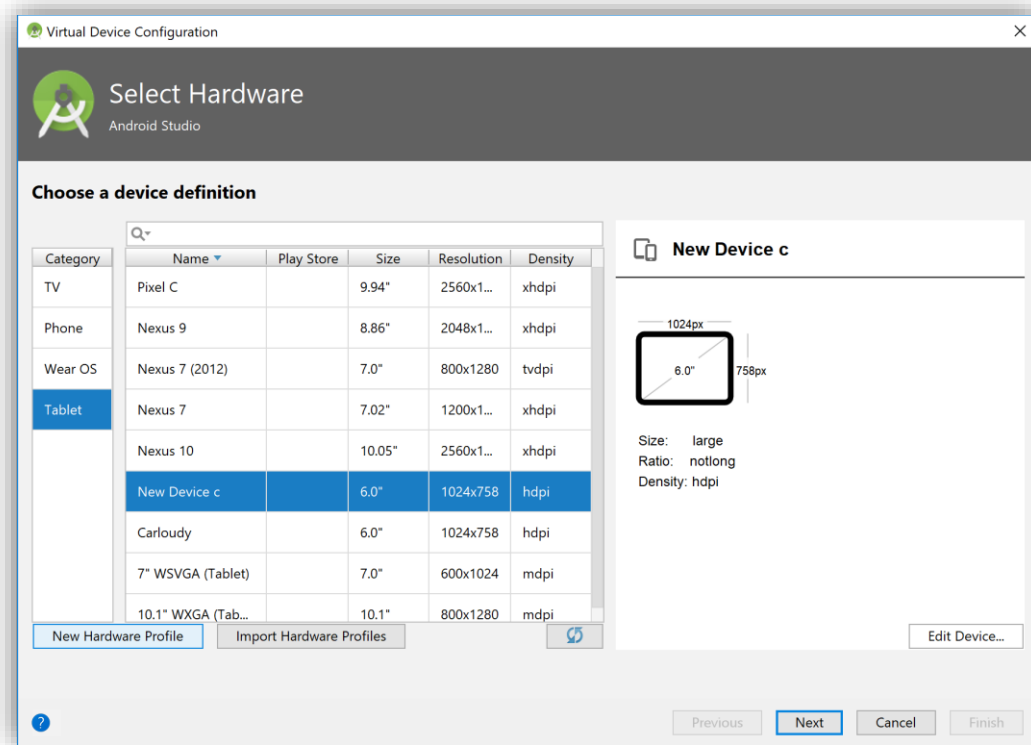
select

+ Create Virtual Device...

. Then the “Virtual Device

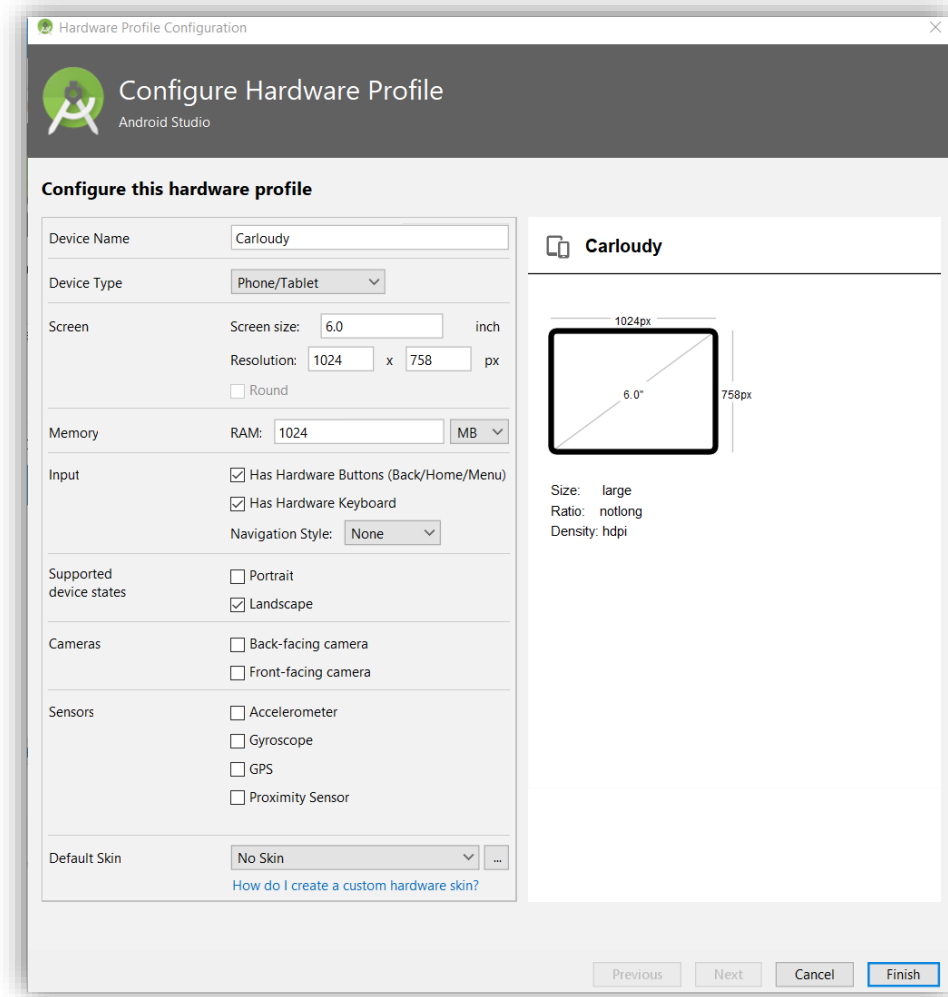
Configuration” window will popup.

4. In “Virtual Device Configuration”, under “Tablet”, add a “New Hardware Profile” and a “Hardware Profile Configuration” window will popup.



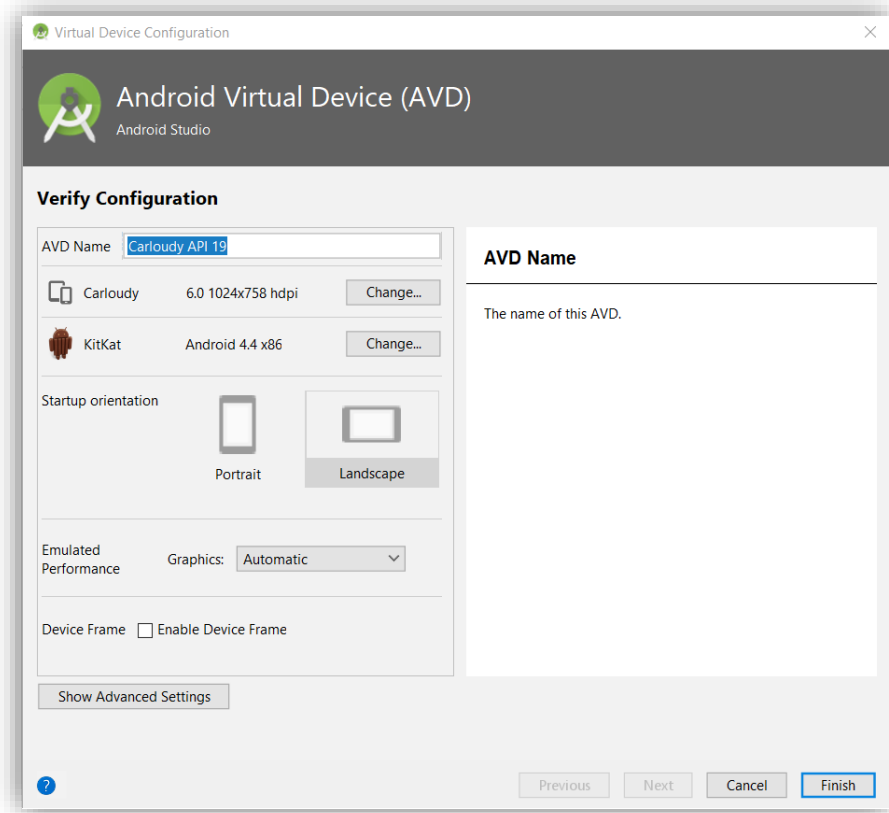
5. In “Hardware Profile Configuration”, set the configurations as below and “Finish”:

- Resolution: 1024 x 758: hdpi
- Screen Size: 6.0
- Landscape: only

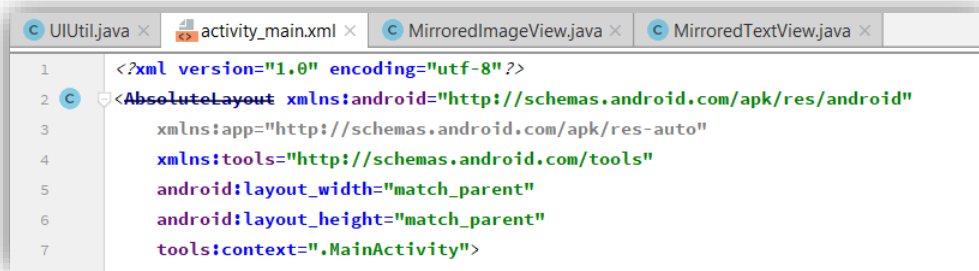


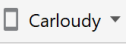
6. Then back to the “[Virtual Device Configuration](#)”, choose the device you just added under “[Tablet](#)” and hit “[Next](#)”. Choose one image you prefer (KitKat API 19 Android 4.4 suggested). After configuration, hit “[Finish](#)”:

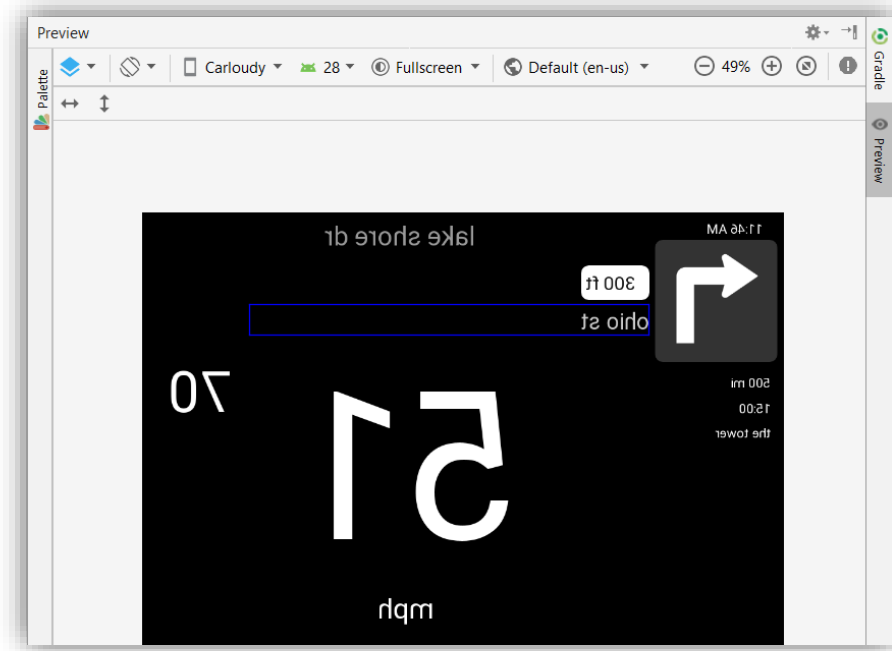
- API: [19](#)
- Target: [Android 4.4](#)
- Startup orientation: [Landscape](#)



7. Go to any activity layout, such as “[activity\\_main.xml](#)” for an easy start.



8. In layout, open “[Preview](#)” on the right side of the screen, under Devices , choose the device name you just added in step 6 starts with “[AVD:](#)”.



9. Still in “Preview”, set the layout theme “AppTheme” to “Black.NoTitleBar.Fullscreen”.
10. Create a new java class named as “UIUtil” directly under the first level of your package. At the beginning of the page, the package should be the one you need to write down.

```

1  package cognitiveai.skin1;
2
3  import android.content.Context;
4  import android.graphics.drawable.Drawable;
5
6  public class UIUtil {
7
8  @  public static Drawable next_distance_bg(Context ctx){
9      return ctx.getResources().getDrawable(R.drawable.next_distance_bg);
10
11  }
12  }

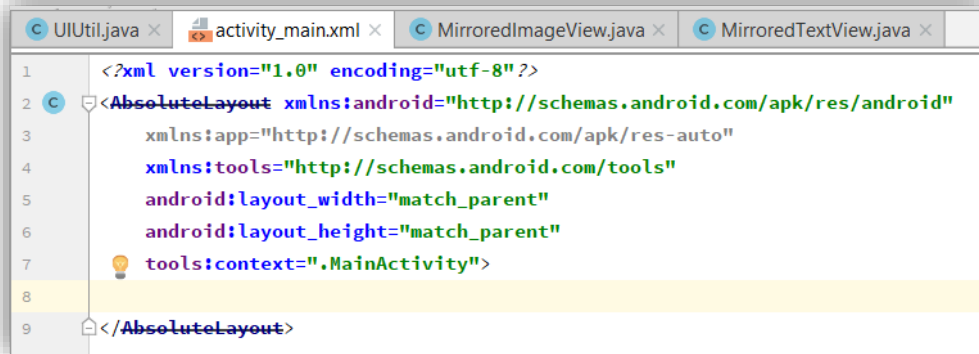
```

11. Create some methods for returning “Drawable” with the same names of the drawables for returning results in resource. For returning the drawable resource “next\_distance\_bg.png”, the method should be

named as “next\_distance\_bg”, omitting the name extension.

```
public static Drawable next_distance_bg(Context ctx){  
    return ctx.getResources().getDrawable(R.drawable.next_distance_bg);  
}
```

12. Change the first level of the frame layout to “[AbsoluteLayout](#)”. (Mostly on the 2<sup>nd</sup> line and remember to change the closing to “[AbsoluteLayout](#)” as well.)



13. Use “[TextView](#)” and “[ImageView](#)” only (We highly recommend using “[MirroredTextView](#)” and “[MirroredImageView](#)” instead. The codes for these two customized widgets will be provided in the “[Help Guides](#)” section.)
14. Make your own layout according to the “[Key Points](#)” in the next section as the guide.
15. After finishing building the skin, build the APK. (In Android Studio, “Build” -> “Build Bundle(s) / APK(s)” -> “Build APK(s)”). And locate the APK file (i.e. [app-debug.apk](#)) under the folder “[...\app\build\outputs\apk\debug](#)”.
16. Locate your layout file (i.e. [activity\\_main.xml](#)) under the folder “[...\app\src\main\res\layout](#)”.
17. Use ADB or Device File Explorer in Android Studio to push the APK file and layout file to the folder “[sdcard](#)”
18. Go to <https://gettingstarted.carloudy.com/> to pair your Android phone with Carloudy.
19. After you connected the Carloudy, go to Carloudy Controller app on your Android phone.
20. In the menu (Swipe right from the left edge of the screen or click the triple line icon on the top left corner of the app.), select “[Skin](#)”.
21. Fill the names you wrote down in step 2.

22. Click “[SEND SIGNAL](#)”, and start Google Maps navigation, you will see your result on Carloudy. Congrats!
23. If you want to stop the customized skin, click “[STOP CUSTOMIZED SKIN](#)”.

## Key Points

### Layout Guide

#### 1. Attributes:

- [id](#)
- [layout\\_x](#)
- [layout\\_y](#)
- [layout\\_width](#)
- [layout\\_height](#)
- [alpha](#)
- [gravity](#) (for TextView / MirroredTextView only)
- [background](#) (for TextView / MirroredTextView only)
- [text](#) (for TextView / MirroredTextView only)
- [textColor](#) (for TextView / MirroredTextView only)
- [textSize](#) (for TextView / MirroredTextView only)
- [src](#) (for ImageView / MirroredImageView only)

#### 2. One line for each attribute

#### 3. “[/>](#)” closing must be on single line itself

#### 4. For TextView / MirroredTextView [ids](#), you need to set certain [id](#) for each text view to function:

- ["tv\\_distance\\_next\\_turn"](#) distance to the next turn
- ["tv\\_next\\_street\\_name"](#) the name of the street of the next turn
- ["tv\\_current\\_street"](#) current street name
- ["tv\\_destination\\_name"](#) destination name
- ["tv\\_eta"](#) estimated time of arrival
- ["tv\\_destination\\_distance"](#) distance to the destination
- ["tv\\_speed\\_limit"](#) speed limit
- ["tv\\_current\\_speed"](#) current speed
- ["tv\\_speed\\_unit"](#) speed unit

- "tv\_clock" clock
5. `layout_width` / `layout_height` unit: `px` / `dp`
  6. `layout_width` / `layout_height` extra definition: `wrap_content` / `match_parent`
  7. `layout_x` / `layout_y` unit: `px` / `dp`
  8. `textSize` unit: `sp` / `dp`
  9. `textColor`: Hex Color code only. Format as "#xxxxxx"
  10. `drawable` resources (including `background` and `src`) must be the same names as your methods in `UIUtil`
  11. no spaces among the characters in each line

## Help Guides

### Helper classes

#### MirroredImageView

```
public class MirroredImageView extends AppCompatActivity {

    public MirroredImageView(Context context) {
        super(context);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.translate(getWidth(), 0);
        canvas.scale(-1, 1);
        super.onDraw(canvas);
    }
}
```

#### MirroredTextView

```
public class MirroredTextView extends AppCompatActivity {

    public MirroredTextView(Context context) {
        super(context);
    }
}
```



```
}  
  
@Override  
protected void onDraw(Canvas canvas) {  
    canvas.translate(getWidth(), 0);  
    canvas.scale(-1, 1);  
    super.onDraw(canvas);  
}  
}
```