3. Change the watch hands

We'll change the hands to give our watch face a more premium makeover.

While the current watch hands get the job done, we want to give the watch hands a more premium makeover. In this section, you will learn more about drawing on Android canvas.

Prepare the paint objects

Before an artist starts, they need to mix their color palette and ready their brush. This is what we will be doing:

- Define the constant `private static final float HAND_END_CAP_RADIUS` within `MyWatchFaceService.Engine` and set it to `4`. Remember to put an `f` after the `4` to indicate that this is a `float` rather than an `int`.
- Define the constant `private static final float SHADOW_RADIUS` within `MyWatchFaceService.Engine` and set it to `6`. Remember the `f`.

Change the `mHandPaint` object with the following parameters in `onCreate`:

`setShadowLayer` with the following parameters:

- radius: SHADOW_RADIUS
- x offset: `0`
- y offset: `0`
- color: `Color.BLACK`
- set the style to `Paint.Style.STROKE` - this means we can draw an outline of shapes

Calculate the length of the watch hands

Different watches may have different size screens. Next to scaling the background, as we did in the last step, we also need to work out the watch hand lengths.

Since screen dimensions do not change, we can insert the calculation code into the method `onSurfaceChanged`:

- hour hand length: `0.5 * width / 2`
- minute hand length: `0.7 * width / 2`
- second hand length: `0.9 * width / 2`

Pro-tip

Remember to put `f` after the decimal number of `0.5`, `0.7` and `0.9` or Java will complain that these are doubles rather than floats and refuse to compile.

Pro-tip

Why do we set the length to be a proportion of the screen rather than half the width minus some fixed number? Proportion scales better for different size screens. Android Wear already has a number of different devices and more will come. This will make it more future proof.

Draw the watch hands

Now that all the length calculation is done, it's time to draw the new watch hands. In this section we will walk you through the code to rotate the watch hands using `canvas.rotate` rather than using sine and cosine. This simplifies the code tremendously reducing the chance for error, allows more complex layouts as we will see later and leaves the optimisation to Android.

First of all we will prepare the canvas for the first hand we draw - the minute hand. So in the onDraw method:

Remove the code which draws the line watch hands using `canvas.drawLine` from the line `canvas.save()` to `canvas.restore()`. We will step you through putting in the new watch hand code in the following steps.

Save the current rotational state of the canvas:

```
canvas.save();
```

Rotate to draw the minute hand:

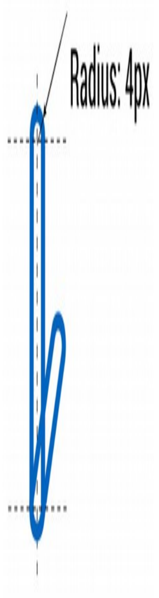
```
canvas.rotate(minutesRotation, mCenterX, mCenterY);
```

The rotate command above means that we have rotated the canvas counter-clockwise by the desired amount. This means we can just draw the minute hand in an upright direction rather than at an angle.

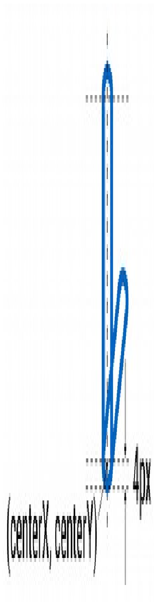
In addition, for our minute hand instead of a line, we want to draw a hollow paper clip like shape for our watch hands. Before we begin, we need to think a little bit about our design:

Hollow paper clip

It's basically a rectangle with two completely round ends. For our design we have chosen 4 px being the radius of the round corners:



Next is to think about the offset that we have to put in in order for the watch hands to be centered correctly. Without this the watch hands will be centered to the bottom of the whole object rather than the center of the bottom round end. It should come as no surprise that this offset is equal to the radius. In our case, this will be 4 px.



Draw the watch hands

Since we need to draw the hour hand and the minute hand in the same way, we create a new private method `drawHand`. This will help us only deal with the `drawRoundRect` code once. If there are any mistakes, we can fix it in one place rather than two. For this method we need both the canvas and the length of the hand:

```
private void drawHand(Canvas canvas, float handLength) {
    canvas.drawRoundRect(mCenterX - HAND_END_CAP_RADIUS,
        mCenterY - handLength, mCenterX + HAND_END_CAP_RADIUS,
        mCenterY + HAND_END_CAP_RADIUS, HAND_END_CAP_RADIUS,
        HAND_END_CAP_RADIUS, mHandPaint);
}
```

Back to the `onDraw` method, we draw the minute hand by calling `drawHand`:

```
drawHand(canvas, mMinuteHandLength);
```

Next, we rotate the canvas to draw the hour hand, taking into account of the rotation of the minute hand drawn before

```
canvas.rotate(hoursRotation - minutesRotation, mCenterX, mCenterY);
```

Draw the hour hand in a similar way to how we draw the minute hand.

Make sure to use the correct hand length for the method calls.

Rotate to draw the second hand, taking into account of the rotation of the hour hand. Do not worry about the rotation that we made for the minute hand. Remember the bit where we rotate **minus** the `minuteRotation`, with that we have "neutralised" the minute hand rotation and all we need to do in this step is to "neutralise" the hour hand rotation.

Lastly, do not worry about the ambient mode code for now the if condition around the second hand. We will put that in in the next section.

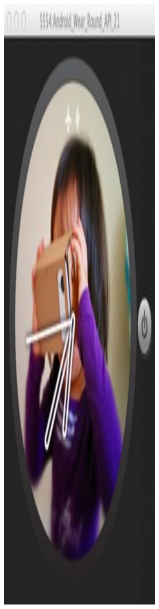
Draw the second hand in the center of the screen adjusting for the radius of the hour and minute watch hand. This means there should be a circular area in the center which is transparent:

```
canvas.drawLine(mCenterX, mCenterY - HAND_END_CAP_RADIUS, mCenterX,
    mCenterY - mSecondHandLength, mHandPaint);
```

Restore the canvas to upright position: `canvas.restore();`

Check your progress

If you run the watch face now, you should see something like this:



Make sure that:

The time is correct - if you are using the emulator and this is not paired with the phone, the time can be different to the time of your laptop. Check the watch face against another watch face or the time that is shown in the log window in Android Studio.

Pro-tip:

If the logging does not appear, try checking four things:

- At the bottom left, the tab "6: Android" is selected.
- Under devices, the correct device is selected. In the example below, it is `Emulator Android_Wear_Round_API_21`.
- The "Log level" is set to "Verbose".
- If nothing shows up, select "No Filters".

The second hand is moving and the minute hand increments at the 60 seconds mark.

It is easy to make mistakes in calculating the rotation angle or setting the center of rotation. If these look odd consider logging these variables and checking the calculation by using `Log.d("MyWatchFaceService", "variableYouWantToLog: " + variableYouWantToLog)` and checking that the value makes sense.

For primitive data types such as `int` or `float`, these can be logged directly as shown. For objects such as `mTime`, you will need to log its `String` representation by logging `mTime.toString()` rather than logging `mTime`. Otherwise, the code will not compile as the method is looking for a String object rather than a Time object.

Pro-tip

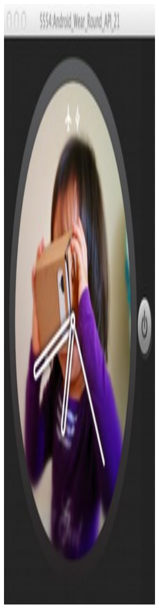
Before publishing your watch face, we recommend that developers test their design in their daily lives and see that it both works correctly technically and that the design works in all circumstances (indoor / outdoor / stationary / on-the-move).

Perfect the center

Coming back to the current design, you might notice that at the center the shadow of the second hand is on top of the hour hand, and it looks a little out of place. A solution is to add a hollow circle in the middle above the second hand to solve this:

```
canvas.drawCircle(mCenterX, mCenterY, HAND_END_CAP_RADIUS, mHandPaint);
```

If you run the watch face again, it should look something like this:



Change the preview image in the watch face selector

In the watch face selector, you can see a preview of the watch face. This can be changed by replacing the preview image `preview_analog.png` in the `res/drawable-nodpi` directory.

Summary

In this step you've learned about:

- Setting up paint objects to draw hollow shapes
- Rotating watch arms into the correct position
- Drawing rounded rectangle shapes.

Next up

Additional to interactive mode, Android Wear watches also have an ambient mode which normally has a more discreet design and is only updated once a minute.

In the next section, we will learn about how to deal with this.