5. Automatic color selection using Palette API

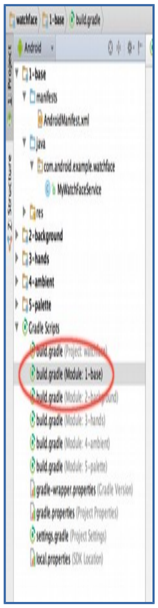
We'll change the hands automatically when a different background comes in.

As part of the Android 5.0 Lollipop launch, we have added a Palette API which helps to automatically select the accent color of a bitmap. We will be using this API for automatically selecting the color of the watch hands. Outside of watch faces, you can use it to generate dynamic color schemes based on the user’s input and fulfil one of the creative visions of material design - a new cross platform design language.

Determine key colors of a bitmap

We will need to initiate the `Palette` object, feed our background bitmap to it and get it to analyse the result:

Open "build.gradle (Module: 1-base)" file under Gradle Scripts:

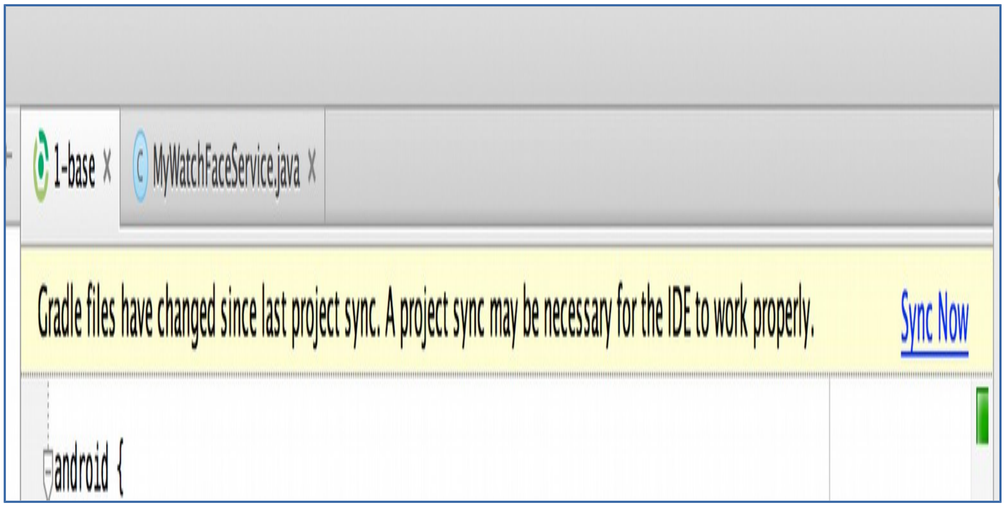


watchface
1-base
build.gradle
Android
1: Project
Z: Structure
1-base
manifests
AndroidManifest.xml
java
com.android.example.watchface
MyWatchFaceService
res
2-background
3-hands
4-ambient
5-palette
Gradle Scripts
build.gradle (Project: watchface)
build.gradle (Module: 1-base)
build.gradle (Module: 2-background)
build.gradle (Module: 3-hands)
build.gradle (Module: 4-ambient)
build.gradle (Module: 5-palette)
gradle-wrapper.properties (Gradle Version)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

Add the following to the `dependencies` section towards the end of the "build.gradle (Module: 1-base)" file:

```
compile 'com.android.support:palette-v7:21.0.0'
```

Press "Sync Now" at the top - this will trigger a rebuild and palette will be available for use.



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
android {
```

Create two `int` variables `mWatchHandColor` and `mWatchHandShadowColor` in `CanvasWatchFaceService.Engine` to store the colors. Initiate them with `Color.WHITE` and `Color.BLACK` respectively.

Instantiate a `Palette` object in `onCreate` after the `mBackgroundBitmap` has been loaded:

```
Palette.generateAsync(mBackgroundBitmap,
    new Palette.PaletteAsyncListener() {
    @Override
    public void onGenerated(Palette palette) {
        if (null != palette) {
            Log.d("onGenerated", palette.toString());
            mWatchHandColor = palette.getVibrantColor(Color.WHITE);
            mWatchHandShadowColor =
                palette.getDarkMutedColor(Color.BLACK);
            setWatchHandColor();
        }
    }
});
```

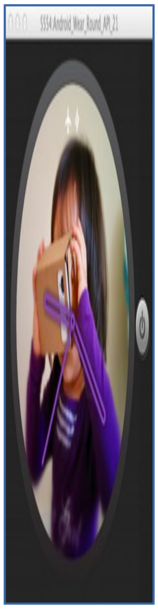
Add a new private method in `MyWatchFaceService.Engine` to set the color of the watch hands:

```
private void setWatchHandColor() {
    if (mAmbient) {
        mHandPaint.setColor(Color.WHITE);
        mHandPaint.setShadowLayer(SHADOW_RADIUS, 0, 0, Color.BLACK);
    } else {
        mHandPaint.setColor(mWatchHandColor);
        mHandPaint.setShadowLayer(SHADOW_RADIUS, 0, 0, mWatchHandShadowColor);
    }
}
```

Call `setWatchHandColor()` method before invalidating the frame in `onAmbientModeChanged`

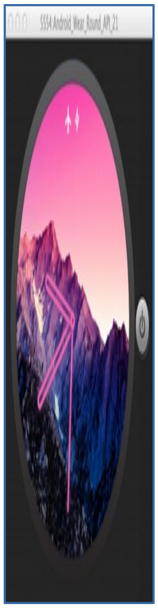
Run the watch face again

Your watch face should look something like this:



Try a different background image

If you copy over the `custom_background2` image from the module *5-palette* to *1-base* and change the `backgroundResId` to `R.drawable.custom_background2` you will see a watch face similar to this:



Summary

In this step you’ve learned about:

- How to select color schemes automatically using the Palette API
- Setting the screen elements color accordingly

This is it for the analog watch face codelab. But wait, there's more to know. Just keep on reading to get inspiration on what to try next.

Next Steps

You can find more information on Android Wear notifications and apps or on watch faces in our developer resources.

Continue customization

Position status icons, the charging icon (the lighting bolt), "Ok Google", etc

- Refer to the `WatchFaceStyle.Builder` documentation for various options from setting gravity to making a semi-transparent background so that they are readable against the watch face.

Put the watch face on the Google Play Store

- Firstly get set up on the Google Play Store if you are not already a developer on the Play Store
- Then, package the watch face with a mobile app
- Lastly, upload the app to the Play Store

Pro-tip:

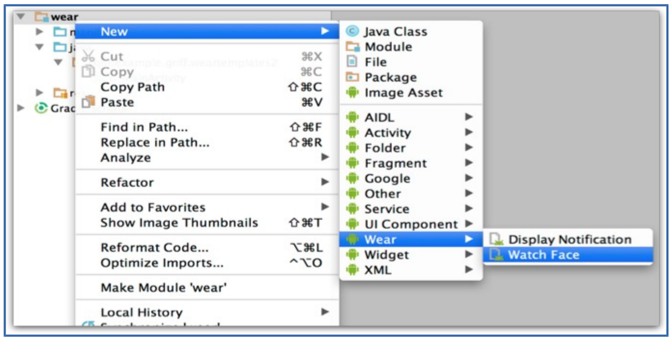
The permissions required for the watch face will need to be copied over to the mobile app.

Create a digital watch face

Use the Android Wear watch face template in Android Studio. Please note that this will require Android Studio version 1.1 or above.

This will help you get the skeleton code:

- Start a new project or open your existing project
- Create a new Android Wear module with no activity if your project does not already have one
- Right-click on the module name on the left hand side
- Select New > Wear > Watch Face



- Select the digital watch template