Kotlin for wearables: use case

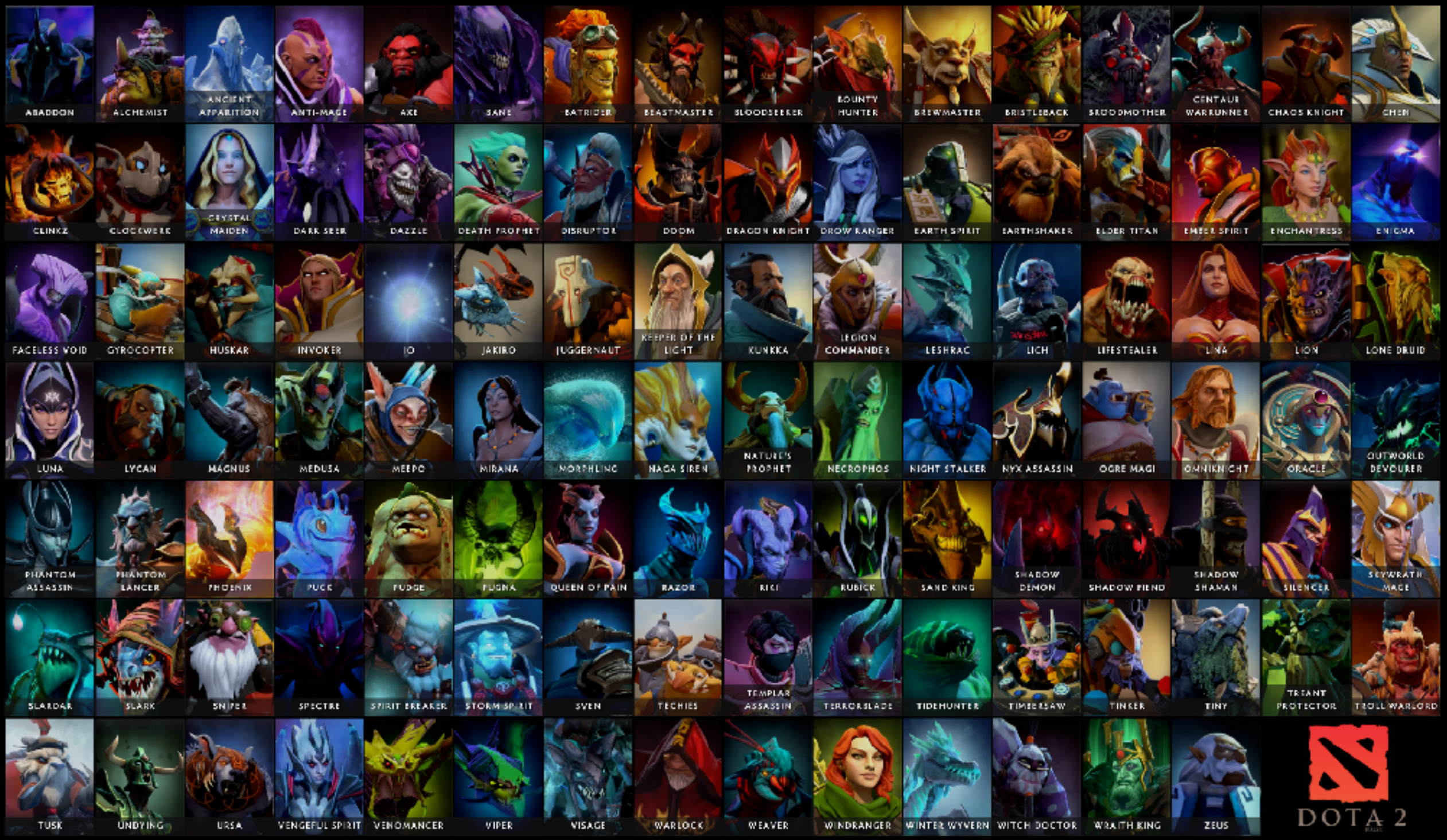
Andrey Mukamolov

Domain

- Dota 2 is a competitive MOBA game
- A lot of fans and money
- Most toxic community

Dota 2

Dota 2 heroes



Dota 2 MMR

	 Herald	 Guardian	 Crusader	 Archon	 Legend	 Ancient	 Divine
	0	840	1680	2520	3360	4200	5040
★	140	980	1820	2660	3500	4340	5180
★★	280	1120	1960	2800	3640	4480	5320
★★★	420	1260	2100	2940	3780	4620	5460
★★★★	560	1400	2240	3080	3920	4760	5600
★★★★★	700	1540	2380	3220	4060	4900	5740


DRINK
VODKA
PLAY
DOTKA


10:15:56
100%
7621


10:18
100%
7621


Dendi
(Данила Кипяток)
7621


70388657
Please specify your Steam ID in the input above so we can fetch your MMR

Wear OS by Google

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="io.github.fobo66.wearmmr">

  <uses-feature android:name="android.hardware.type.watch"/>

  <uses-permission android:name="android.permission.WAKE_LOCK"/>
  <uses-permission android:name="android.permission.INTERNET" />
  …
  <uses-permission
android:name="com.google.android.wearable.permission.RECEIVE_COMPLICATION_DATA"/>

  <application
      android:name=".App"
      …
      android:theme="@android:style/Theme.DeviceDefault">
    <uses-library
        android:name="com.google.android.wearable"
        android:required="true"/>
    <meta-data
        android:name="com.google.android.wearable.standalone"
        android:value="true"/>
```

Wear OS by Google

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="io.github.fobo66.wearmmr">

  <uses-feature android:name="android.hardware.type.watch"/>

  <uses-permission android:name="android.permission.WAKE_LOCK"/>
  <uses-permission android:name="android.permission.INTERNET" />
  …
  <uses-permission
android:name="com.google.android.wearable.permission.RECEIVE_COMPLICATION_DATA"/>

  <application
      android:name=".App"
      …
      android:theme="@android:style/Theme.DeviceDefault">
    <uses-library
        android:name="com.google.android.wearable"
        android:required="true"/>
    <meta-data
        android:name="com.google.android.wearable.standalone"
        android:value="true"/>
```

Wear OS by Google

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="io.github.fobo66.wearmmr">

  <uses-feature android:name="android.hardware.type.watch"/>

  <uses-permission android:name="android.permission.WAKE_LOCK"/>
  <uses-permission android:name="android.permission.INTERNET" />
  …
  <uses-permission
android:name="com.google.android.wearable.permission.RECEIVE_COMPLICATION_DATA"/>

  <application
      android:name=".App"
      …
      android:theme="@android:style/Theme.DeviceDefault">
    <uses-library
        android:name="com.google.android.wearable"
        android:required="true"/>
    <meta-data
        android:name="com.google.android.wearable.standalone"
        android:value="true"/>
```

Wear OS by Google

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="io.github.fobo66.wearmmr">

  <uses-feature android:name="android.hardware.type.watch"/>

  <uses-permission android:name="android.permission.WAKE_LOCK"/>
  <uses-permission android:name="android.permission.INTERNET" />
  …
  <uses-permission
android:name="com.google.android.wearable.permission.RECEIVE_COMPLICATION_DATA"/>

  <application
      android:name=".App"
      …
      android:theme="@android:style/Theme.DeviceDefault">
    <uses-library
        android:name="com.google.android.wearable"
        android:required="true"/>
    <meta-data
        android:name="com.google.android.wearable.standalone"
        android:value="true"/>
```

Wear OS by Google

```
android {
    //…
    minSdkVersion 23 // 25 for standalone
    targetSdkVersion 27
}

dependencies {

    compileOnly "com.google.android.wearable:wearable:$wearableVersion"

    // …
}
```

Why Kotlin?

- Maintainability
- Readability
- Convenience (Canvas)

Same as regular Android apps

```
class MainActivity : WearableActivity() {

    val db: MatchmakingDatabase by inject()

    @BindView(R.id.bottom_action_drawer)

    lateinit var navigationDrawer: WearableActionDrawerView


    @BindView(R.id.player_pic)

    lateinit var playerPic: ImageView


    @BindView(R.id.player_name)

    lateinit var playerName: TextView

// …
```

Same as regular Android apps

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        ButterKnife.bind(this)

        // Enables Always-on
        setAmbientEnabled()
        //…
}
```

DI

```
val databaseModule = applicationContext {
    bean {
        Room.databaseBuilder(
            androidApplication(),
            MatchmakingDatabase::class.java,
            "matchmaking"
        )
        .build()
    }
}
```

DI

```
val apiModule = applicationContext {
    bean { provideHttpClient() }
    bean { provideApiClient(get()) }
}
fun provideHttpClient(): OkHttpClient {
    return OkHttpClient()
}
fun provideApiClient(httpClient: OkHttpClient): MatchmakingRatingApi {
    val retrofit: Retrofit = Retrofit.Builder()
        .baseUrl(API_BASE_URL)
        .addConverterFactory(JacksonConverterFactory.create(jacksonObjectMapper()))
        .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
        .client(httpClient)
        .build()
    return retrofit.create(MatchmakingRatingApi::class.java)
}
```

Ambient mode

```
        /* Whether the display supports fewer bits for each color in
ambient mode. When true, we disable anti-aliasing in ambient mode*/

private var lowBitAmbient: Boolean = false

private var burnInProtection: Boolean = false

private var modeAmbient: Boolean = false
```

Ambient mode

```
override fun onPropertiesChanged(properties: Bundle) {
    super.onPropertiesChanged(properties)
    lowBitAmbient = properties.getBoolean(
        WatchFaceService.PROPERTY_LOW_BIT_AMBIENT, false
    )
    burnInProtection = properties.getBoolean(
        WatchFaceService.PROPERTY_BURN_IN_PROTECTION, false
    )
}
```

Ambient mode

```
override fun onAmbientModeChanged(inAmbientMode: Boolean) {

        super.onAmbientModeChanged(inAmbientMode)

        modeAmbient = inAmbientMode

        if (lowBitAmbient) {

            textPaint.isAntiAlias = !inAmbientMode

        }

        batteryComplication.setInAmbientMode(inAmbientMode)

        ratingComplication.setInAmbientMode(inAmbientMode)

}
```

Watchface

```
override fun onDraw(canvas: Canvas, bounds: Rect) {
            // Draw the background.
            if (modeAmbient) {
                canvas.drawColor(Color.BLACK)
            } else {
                canvas.drawBitmap(
                    backgroundBitmap,
                    0f, 0f,
                    backgroundPaint
                )
            }
// …
        }
```

Watchface

```
// Draw H:MM in ambient mode or H:MM:SS in interactive mode.
            val now = System.currentTimeMillis()
            calendar.timeInMillis = now

            val time = if (modeAmbient)
                String.format(
                    "%d:%02d", calendar.get(Calendar.HOUR),
                    calendar.get(Calendar.MINUTE)
                )
            else
                String.format(
                    "%d:%02d:%02d", calendar.get(Calendar.HOUR),
                    calendar.get(Calendar.MINUTE),
calendar.get(Calendar.SECOND)
                )
            canvas.drawText(time, timeXOffset, timeYOffset,
textPaint)
```

Watchface

```
batteryComplication.draw(canvas, now)
ratingComplication.draw(canvas, now)
```

Complications

```
    <service
        android:name=".RatingComplicationProviderService"
        android:icon="@drawable/ic_rating"
        android:label="@string/complications_provider_matchmaking_rating_name"

android:permission="com.google.android.wearable.permission.BIND_COMPLICATION_PROVIDER">
      <meta-data
          android:name="android.support.wearable.complications.SUPPORTED_TYPES"
          android:value="SHORT_TEXT"/>
      <meta-data
          android:name="android.support.wearable.complications.UPDATE_PERIOD_SECONDS"
          android:value="43200"/>
    <meta-data
        android:name="android.support.wearable.complications.SAFE_WATCH_FACES"
        android:value="io.github.fobo66.wearmmr.MatchmakingRatingWatchFace"/>
      <intent-filter>
        <action
android:name="android.support.wearable.complications.ACTION_COMPLICATION_UPDATE_REQUEST
"/>
      </intent-filter>
    </service>
```

Complications

```
    <service
        android:name=".RatingComplicationProviderService"
        android:icon="@drawable/ic_rating"
        android:label="@string/complications_provider_matchmaking_rating_name"

android:permission="com.google.android.wearable.permission.BIND_COMPLICATION_PROVIDER">
        <meta-data
            android:name="android.support.wearable.complications.SUPPORTED_TYPES"
            android:value="SHORT_TEXT"/>
        <meta-data
            android:name="android.support.wearable.complications.UPDATE_PERIOD_SECONDS"
            android:value="43200"/>
    <meta-data
        android:name="android.support.wearable.complications.SAFE_WATCH_FACES"
        android:value="io.github.fobo66.wearmmr.MatchmakingRatingWatchFace"/>
        <intent-filter>
          <action
android:name="android.support.wearable.complications.ACTION_COMPLICATION_UPDATE_REQUEST
"/>
        </intent-filter>
    </service>
```

Complications

```
    <service
        android:name=".RatingComplicationProviderService"
        android:icon="@drawable/ic_rating"
        android:label="@string/complications_provider_matchmaking_rating_name"

android:permission="com.google.android.wearable.permission.BIND_COMPLICATION_PROVIDER">
        <meta-data
            android:name="android.support.wearable.complications.SUPPORTED_TYPES"
            android:value="SHORT_TEXT"/>
        <meta-data
            android:name="android.support.wearable.complications.UPDATE_PERIOD_SECONDS"
            android:value="43200"/>
    <meta-data
        android:name="android.support.wearable.complications.SAFE_WATCH_FACES"
        android:value="io.github.fobo66.wearmmr.MatchmakingRatingWatchFace"/>
        <intent-filter>
          <action
android:name="android.support.wearable.complications.ACTION_COMPLICATION_UPDATE_REQUEST
"/>
        </intent-filter>
    </service>
```

Complications

```
    <service
        android:name=".RatingComplicationProviderService"
        android:icon="@drawable/ic_rating"
        android:label="@string/complications_provider_matchmaking_rating_name"

android:permission="com.google.android.wearable.permission.BIND_COMPLICATION_PROVIDER">
        <meta-data
            android:name="android.support.wearable.complications.SUPPORTED_TYPES"
            android:value="SHORT_TEXT"/>
        <meta-data
            android:name="android.support.wearable.complications.UPDATE_PERIOD_SECONDS"
            android:value="43200"/>
    <meta-data
        android:name="android.support.wearable.complications.SAFE_WATCH_FACES"
        android:value="io.github.fobo66.wearmmr.MatchmakingRatingWatchFace"/>
        <intent-filter>
          <action
android:name="android.support.wearable.complications.ACTION_COMPLICATION_UPDATE_REQUEST
"/>
        </intent-filter>
    </service>
```

Complications

```
setActiveComplications(BATTERY_PROVIDER_ID,
   RATING_PROVIDER_ID)

setDefaultSystemComplicationProvider(
            BATTERY_PROVIDER_ID,
            SystemProviders.WATCH_BATTERY, TYPE_SHORT_TEXT
        )
        setDefaultComplicationProvider(
            RATING_PROVIDER_ID,
            ComponentName(applicationContext,
RatingComplicationProviderService::class.java),
            TYPE_SHORT_TEXT
        )
```

Complications

```
setActiveComplications(BATTERY_PROVIDER_ID,
   RATING_PROVIDER_ID)

setDefaultSystemComplicationProvider(
            BATTERY_PROVIDER_ID,
            SystemProviders.WATCH_BATTERY, TYPE_SHORT_TEXT
        )
        setDefaultComplicationProvider(
            RATING_PROVIDER_ID,
            ComponentName(applicationContext,
RatingComplicationProviderService::class.java),
            TYPE_SHORT_TEXT
        )
```

Complications

```
override fun onComplicationDataUpdate(
    watchFaceComplicationId: Int,
    data: ComplicationData?
) {
            when (watchFaceComplicationId) {
                BATTERY_PROVIDER_ID ->
batteryComplication.setComplicationData(data)
                RATING_PROVIDER_ID ->
ratingComplication.setComplicationData(data)
            }
            invalidate()
        }
```

Complications

```
val batteryComplicationBounds = Rect(
    horizontalOffset,
    verticalOffset,
    (horizontalOffset + sizeOfComplication),
    (verticalOffset + sizeOfComplication)
)

batteryComplication.bounds = batteryComplicationBounds

val ratingComplicationBounds = Rect(
    (midpointOfScreen + horizontalOffset),
    verticalOffset,
    (midpointOfScreen + horizontalOffset + sizeOfComplication),
    (verticalOffset + sizeOfComplication)
)

ratingComplication.bounds = ratingComplicationBounds
```

Complications

```
val batteryComplicationBounds = Rect(
    horizontalOffset,
    verticalOffset,
    (horizontalOffset + sizeOfComplication),
    (verticalOffset + sizeOfComplication)
)

batteryComplication.bounds = batteryComplicationBounds
```

WTF?

```
val ratingComplicationBounds = Rect(
    (midpointOfScreen + horizontalOffset),
    verticalOffset,
    (midpointOfScreen + horizontalOffset + sizeOfComplication),
    (verticalOffset + sizeOfComplication)
)

ratingComplication.bounds = ratingComplicationBounds
```

Complications

```
class RatingComplicationProviderService :
ComplicationProviderService() {

    override fun onComplicationUpdate(

        complicationId: Int, dataType: Int,
complicationManager: ComplicationManager?

    ) {

        updateRating(complicationManager, complicationId)

    }

    // …

}
```

Complications

```
val complicationData: ComplicationData = Builder(ComplicationData.TYPE_SHORT_TEXT)
                        .setIcon(
                              Icon.createWithResource(
                                 applicationContext,
                                    drawable.ic_rating
                                 )
                           )
                        .setShortText(ComplicationText.plainText(it.toString()))
                        .setImageContentDescription(
                           ComplicationText.plainText(«Rating value»)
                        )
                        .build()
```

Complications

```
<drawable xmlns:app="http://schemas.android.com/apk/res-auto"
    class="android.support.wearable.complications.rendering.ComplicationDrawable"
    app:backgroundColor="#f44336"
    app:borderColor="#f44336"
    app:iconColor="#e0e0e0"
    app:textColor="#e0e0e0"
    app:titleColor="#e0e0e0">
    <ambient
        app:backgroundColor="@android:color/transparent"
        app:iconColor="#757575"
        app:textColor="#bdbdbd" />
</drawable>
```

Tips and tricks

- Handle ambient always
- Carefully analyze your use case for wear app
- It's still a watch
- Don't overload watchface with complications

https://github.com/fobo66/WearMMR

Summary

Thanks!