ウォッチフェイス作ってみた

shibuya.apk #41 @Kaito-Dogi

自己紹介

- どぎー
- 株式会社ゆめみ
- Androidエンジニア
- Pixel Watch 愛用中

@Kaito_Dogi

@Kaito-Dogi

スマートウォッチ使ってますか！

画像：https://store.google.com/jp/product/google_pixel_watch?hl=ja

毎日使ってるのに味気ない…

自分だけのカスタマイズをしたい…

ウォッチフェイスで実現できる！

ウォッチフェイスとは？

- 「文字盤」
- ユーザーが自身を表現できる
  分かりやすい手段
- データソースからのデータを
  表示できる（追加機能）
- タップイベントの処理

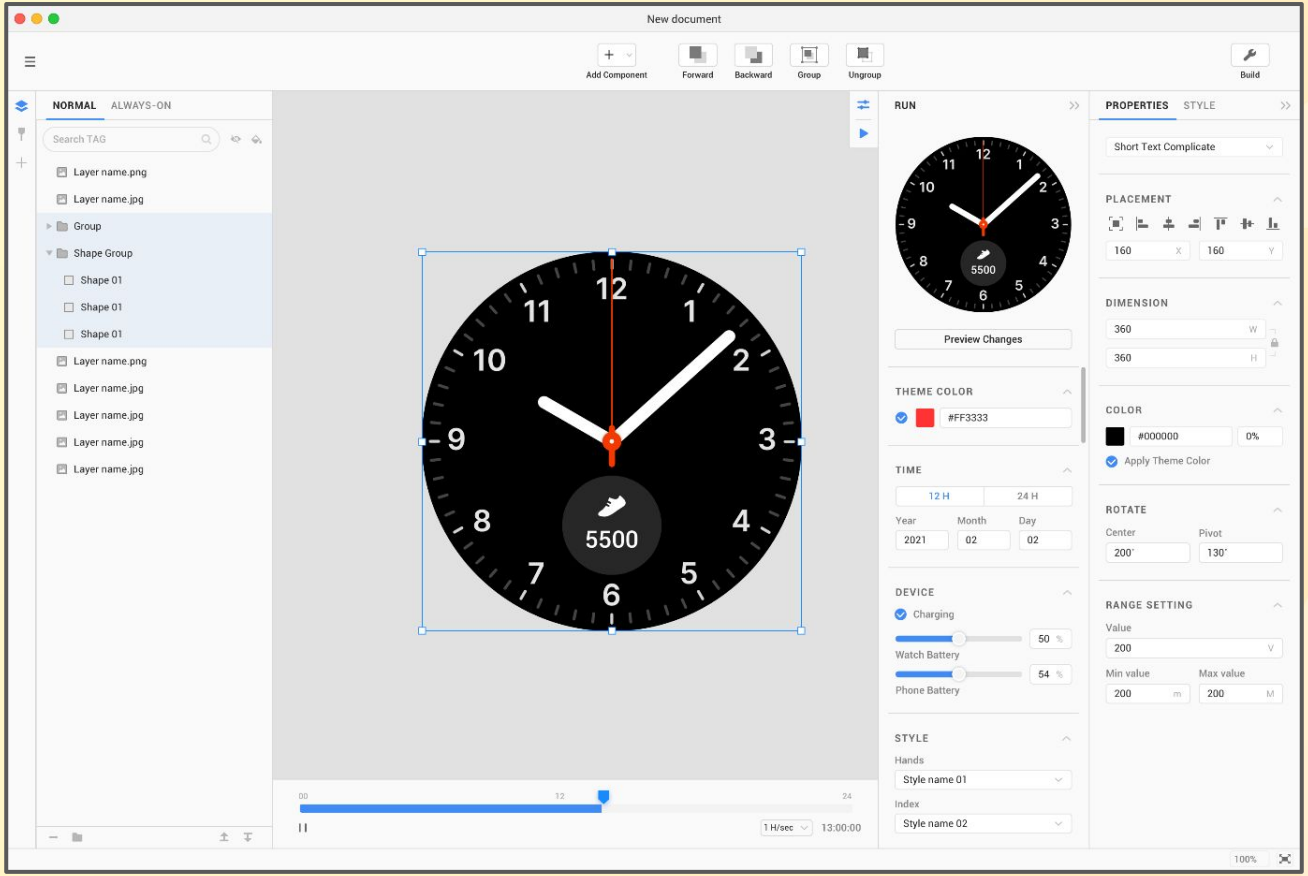
ウォッチフェイスを
自分でも作りたい！

ウォッチフェイスは手軽に作れる！

そう、WatchFaceStudioならね

WatchFaceStudioとは？

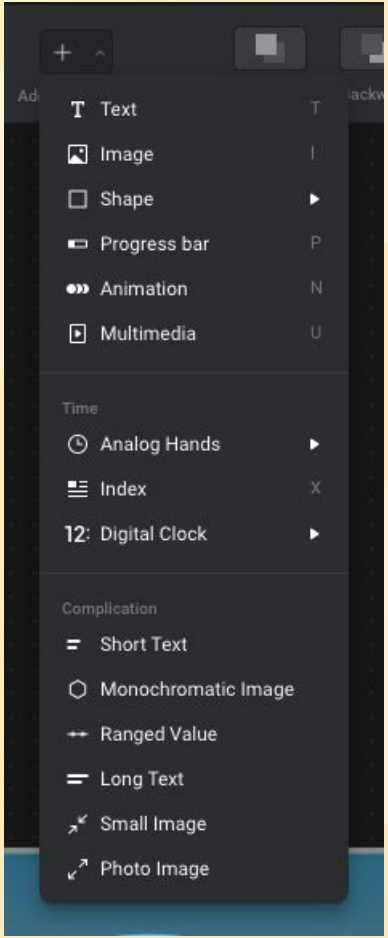
- 「ウォッチフェイスデザインツール」
- コーディング不要！
- Google Play に公開可能
- Samsung が作成
- https://developer.samsung.com/watch-face-studio/download.html



画像：https://developer.android.com/training/wearables/wfs

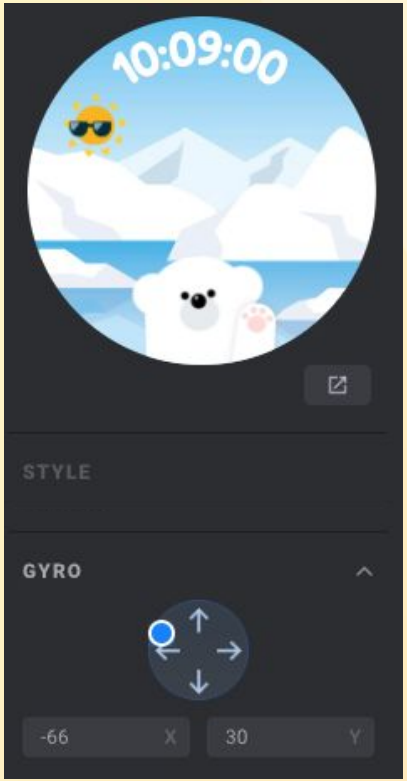
コンポーネント

- テキスト
  - 静的な標準テキスト
  - 動的なデータテキスト（タグ式を使用）
- 図形（楕円・長方形など）
- 画像・アニメーション
- AGIF・Lottie・WebP の再生（マルチメディア）
- アナログ時計・デジタル時計
- プログレスバー



ジャイロ

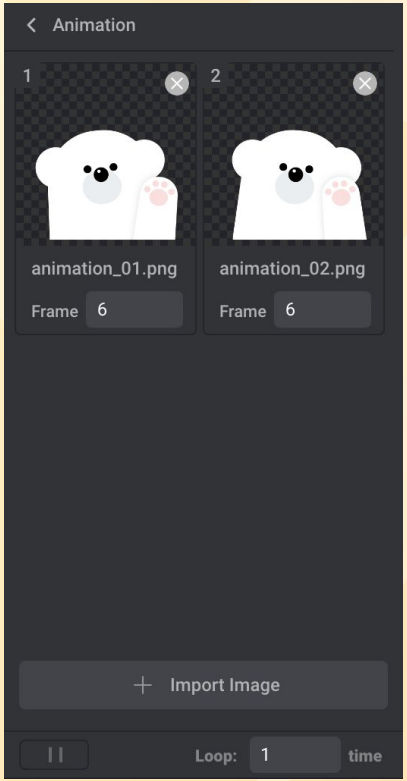
- 端末の傾きに応じて以下を変更できる
  - 大きさ
  - 位置
  - 傾き
  - 不透明度



アニメーション

- ❏ 複数の画像をパラパラマンガのように再生
- ❏ 画像ごとに Frame を設定
- ❏ メイン画面で再生して確認可能

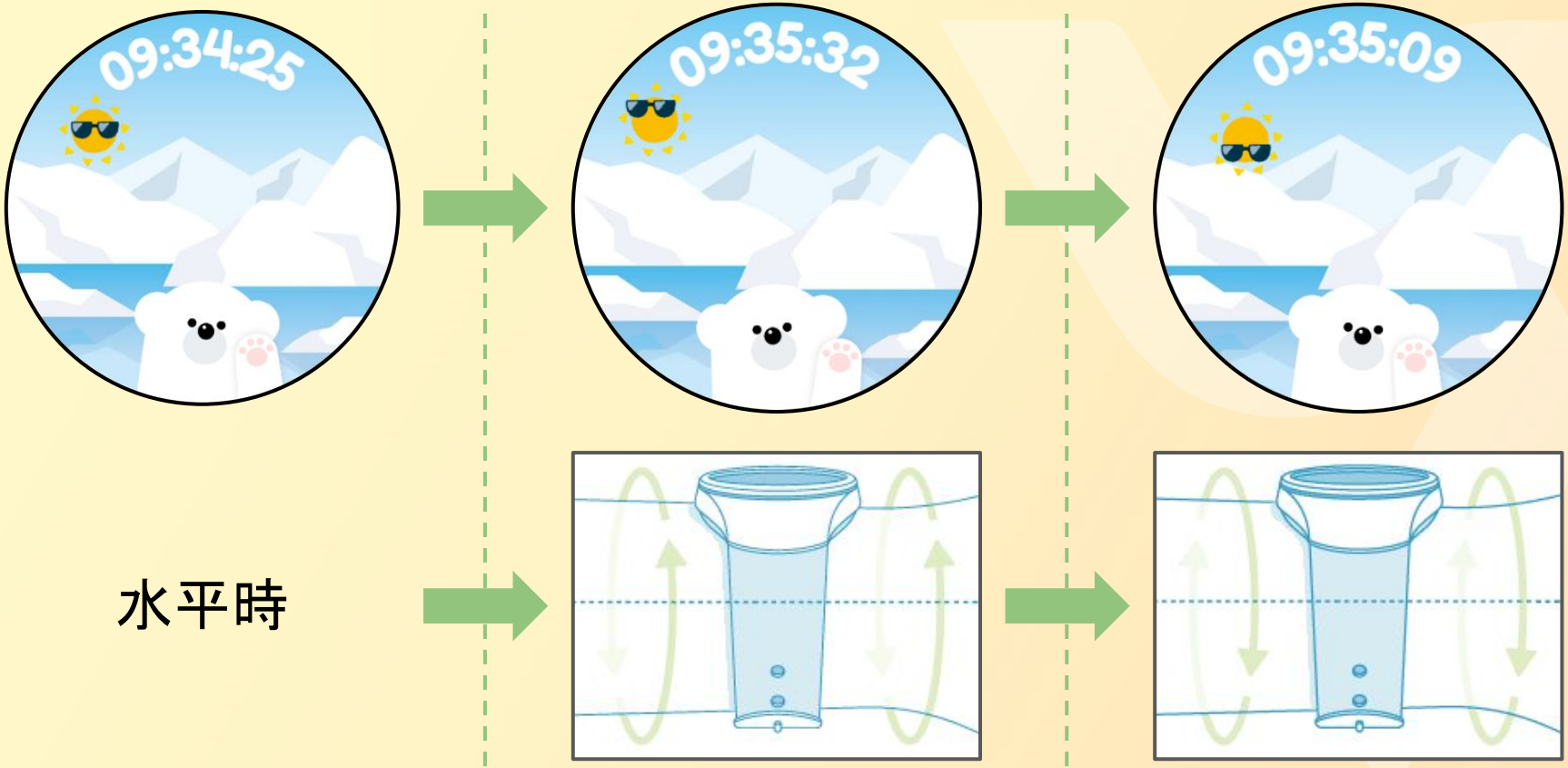
- ❏ AGIF・Lottie・WebP のような、
  既にアニメーション化されている画像は
  マルチメディアで再生！



実際に作ってみた！

- タグ式で時刻を表示
  ([HOUR_1_12_Z]):([MIN_Z]):([SEC_Z])

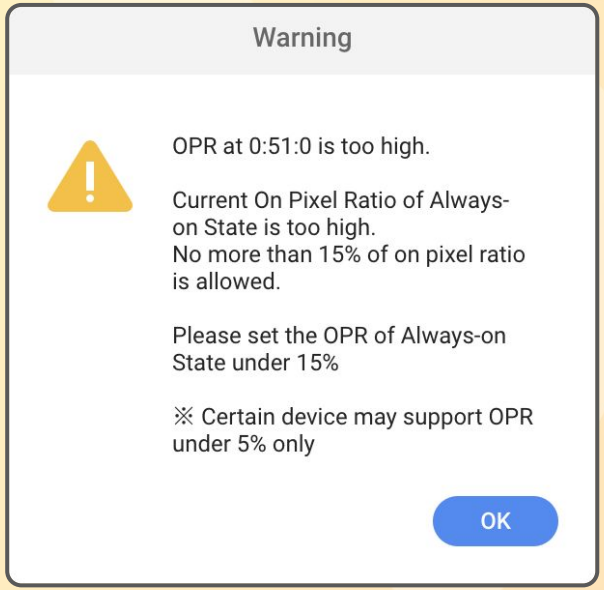
  01~12　00~59　00~59
- デジタル時計をカーブ
- 太陽くんをジャイロで動かす
- 白くまくんをアニメーションで動かす



手首：https://developer.android.com/training/wearables/wfs

注意すること

- 常時オン状態で使用できるピクセルのうち15%以下を使用する設計を推奨
- 大幅に超えると拒否される！
- 常時オン状態は自動的に生成されるので自分で調整

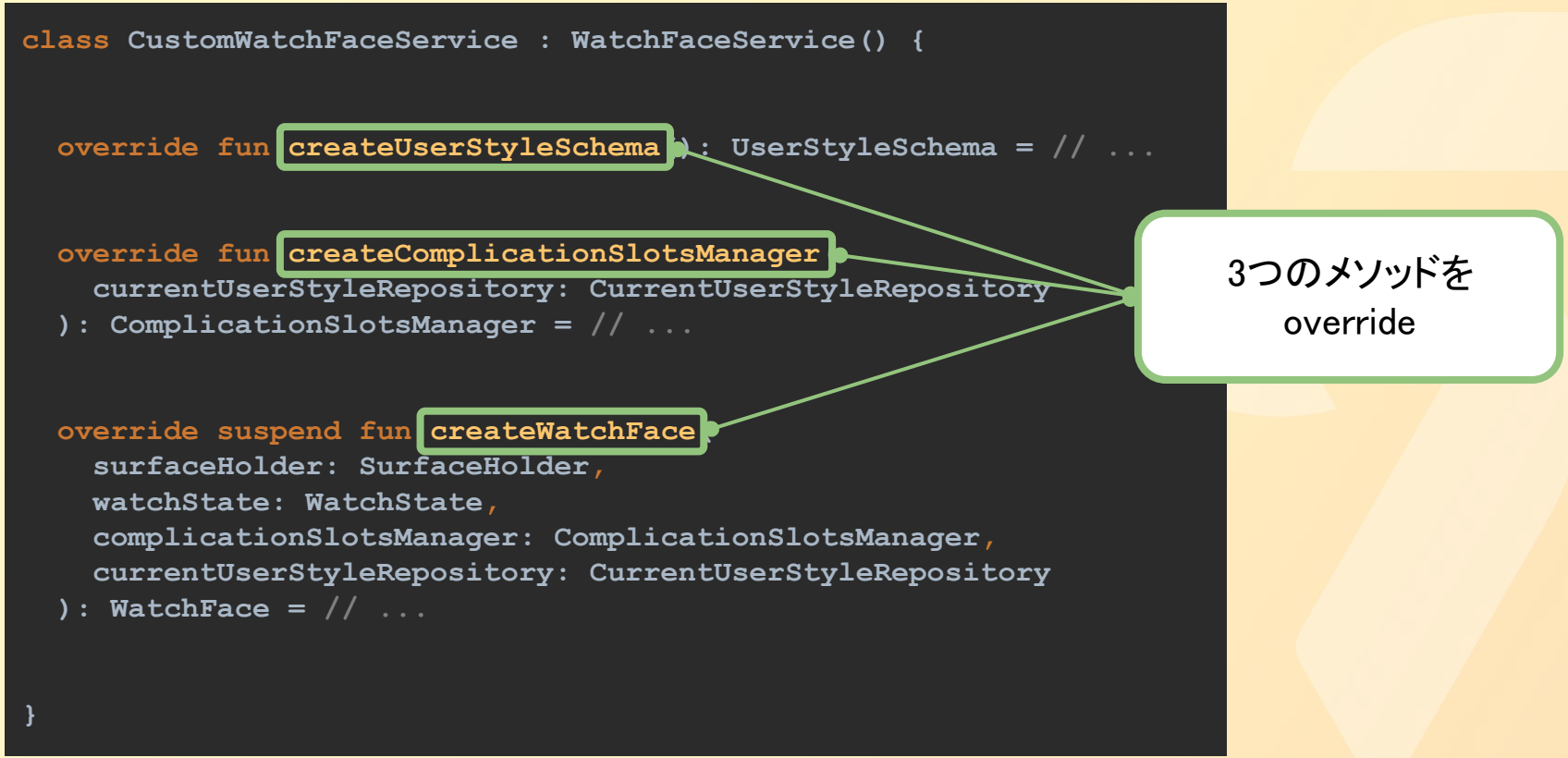


ウォッチフェイスのコードを
眺めてみる👀

```
class CustomWatchFaceService : WatchFaceService() {


  override fun createUserStyleSchema(): UserStyleSchema = // ...


  override fun createComplicationSlotsManager(
    currentUserStyleRepository: CurrentUserStyleRepository
  ): ComplicationSlotsManager = // ...


  override suspend fun createWatchFace(
    surfaceHolder: SurfaceHolder,
    watchState: WatchState,
    complicationSlotsManager: ComplicationSlotsManager,
    currentUserStyleRepository: CurrentUserStyleRepository
  ): WatchFace = // ...


}
```

```
class CustomWatchFaceService : WatchFaceService() {

  override fun createUserStyleSchema(): UserStyleSchema = // ...

  override fun createComplicationSlotsManager(
    currentUserStyleRepository: CurrentUserStyleRepository
  ): ComplicationSlotsManager = // ...

  override suspend fun createWatchFace(
    surfaceHolder: SurfaceHolder,
    watchState: WatchState,
    complicationSlotsManager: ComplicationSlotsManager,
    currentUserStyleRepository: CurrentUserStyleRepository
  ): WatchFace = // ...

}
```

WatchFaceService
を継承


class CustomWatchFaceService : WatchFaceService() {
override fun createUserStyleSchema(): UserStyleSchema = // ...
override fun createComplicationSlotsManager(
currentUserStyleRepository: CurrentUserStyleRepository
): ComplicationSlotsManager = // ...
override suspend fun createWatchFace(
surfaceHolder: SurfaceHolder,
watchState: WatchState,
complicationSlotsManager: ComplicationSlotsManager,
currentUserStyleRepository: CurrentUserStyleRepository
): WatchFace = // ...
}
3つのメソッドを
override

```
class CustomWatchFaceService : WatchFaceService() {

  override fun createUserStyleSchema(): UserStyleSchema = // ...

  override fun createComplicationSlotsManager(
    currentUserStyleRepository: CurrentUserStyleRepository
  ): ComplicationSlotsManager = // ...

  override suspend fun createWatchFace(
    surfaceHolder: SurfaceHolder,
    watchState: WatchState,
    complicationSlotsManager: ComplicationSlotsManager,
    currentUserStyleRepository: CurrentUserStyleRepository
  ): WatchFace = // ...

}
```

スキーマの設定
（色・分針の長さなど）

```
class CustomWatchFaceService : WatchFaceService() {


  override fun createUserStyleSchema(): UserStyleSchema = // ...


  override fun createComplicationSlotsManager(
    currentUserStyleRepository: CurrentUserStyleRepository
  ): ComplicationSlotsManager = // ...


  override suspend fun createWatchFace(
    surfaceHolder: SurfaceHolder,
    watchState: WatchState,
    complicationSlotsManager: ComplicationSlotsManager,
    currentUserStyleRepository: CurrentUserStyleRepository
  ): WatchFace = // ...


}
```

Complication スロットの初期化

```
class CustomWatchFaceService : WatchFaceService() {


  override fun createUserStyleSchema(): UserStyleSchema = // ...


  override fun createComplicationSlotsManager(
    currentUserStyleRepository: CurrentUserStyleRepository
  ): ComplicationSlotsManager = // ...


  override suspend fun createWatchFace(
    surfaceHolder: SurfaceHolder,
    watchState: WatchState,
    complicationSlotsManager: ComplicationSlotsManager,
    currentUserStyleRepository: CurrentUserStyleRepository
  ): WatchFace = // ...


}
```

ウォッチフェイスを
レンダリング（描画）

良きスマートウォッチライフを😙

参考記事

- Build watch faces | Android Developers
  https://developer.android.com/training/wearables/watch-faces
- Watch Face Studio | Android Developers
  https://developer.android.com/training/wearables/wfs
- **WatchFace Sample (Kotlin)**
  https://github.com/android/wear-os-samples/tree/main/WatchFaceKotlin
- Watch Face Studio | Samsung Developers
  https://developer.samsung.com/watch-face-studio/overview.html
- DroidKaigi 2022 – Introduction to Wear OS Application Development | Ryo Yamazaki
  https://youtu.be/o0HhsnVAGNA
- Watch Face Studio でウォッチフェイスを作ろう
  https://funnelbit.hatenablog.com/entry/2022/11/28/150138
- [Android] Watch Faceをささっと作成
  https://cocoamix.jp/archives/9482

ありがとうございました🙌