Bluetooth Low Energy

Hengyu Wu, Shaohan Xu, Yuanzhen Lin

WHAT IS BLUETOOTH LOW ENERGY (BLE)?

Bluetooth®

The global standard for simple, secure device communication and positioning

Bluetooth® Classic

Solution Areas

AUDIO STREAMING

DATA TRANSFER

Device Communication

POINT-TO-POINT

Basic Rate/ Enhanced Data Rate Radio

Bluetooth® Low Energy

Solution Areas

AUDIO STREAMING
(COMING)

DATA TRANSFER

LOCATION SERVICES

DEVICE NETWORKS

Device Communication

POINT-TO-POINT

BROADCAST

MESH

Device Positioning

PRESENCE

PROXIMITY

DIRECTION

DISTANCE
(COMING)

Low Energy Radio

Find BLE devices

Scan devices

1. Filter
2. Scan setting
3. Scan callback

Filter

1. Uuid
2. Device name
3. Mac address

Scan Setting

- Scan Mode
  a. SCAN_MODE_LOW_POWER
  b. SCAN_MODE_BALANCED
  c. SCAN_MODE_LOW_LATENCY
  d. SCAN_MODE_OPPORTUNISTIC

```
ScanSettings scanSettings = new ScanSettings.Builder()
        .setScanMode(ScanSettings.SCAN_MODE_LOW_POWER)
        .setCallbackType(ScanSettings.CALLBACK_TYPE_ALL_MATCHES)
        .setMatchMode(ScanSettings.MATCH_MODE_AGGRESSIVE)
        .setNumOfMatches(ScanSettings.MATCH_NUM_ONE_ADVERTISEMENT)
        .setReportDelay(0L)
        .build();
```

Scan Setting

- Callback type
  a. CALLBACK_TYPE_ALL_MATCHES
  b. CALLBACK_TYPE_FIRST_MATCH
  c. CALLBACK_TYPE_MATCH_LOST

```
ScanSettings scanSettings = new ScanSettings.Builder()
        .setScanMode(ScanSettings.SCAN_MODE_LOW_POWER)
        .setCallbackType(ScanSettings.CALLBACK_TYPE_ALL_MATCHES)
        .setMatchMode(ScanSettings.MATCH_MODE_AGGRESSIVE)
        .setNumOfMatches(ScanSettings.MATCH_NUM_ONE_ADVERTISEMENT)
        .setReportDelay(0L)
        .build();
```

Scan Setting

- Match mode
  a. MATCH_MODE_AGGRESSIVE
  b. MATCH_MODE_STICKY

```
ScanSettings scanSettings = new ScanSettings.Builder()
        .setScanMode(ScanSettings.SCAN_MODE_LOW_POWER)
        .setCallbackType(ScanSettings.CALLBACK_TYPE_ALL_MATCHES)
        .setMatchMode(ScanSettings.MATCH_MODE_AGGRESSIVE)
        .setNumOfMatches(ScanSettings.MATCH_NUM_ONE_ADVERTISEMENT)
        .setReportDelay(0L)
        .build();
```

Scan Setting

- Number of matches
  a. MATCH_NUM_ONE_ADVERTISEMENT
  b. MATCH_NUM_FEW_ADVERTISEMENT
  c. MATCH_NUM_MAX_ADVERTISEMENT

```
ScanSettings scanSettings = new ScanSettings.Builder()
        .setScanMode(ScanSettings.SCAN_MODE_LOW_POWER)
        .setCallbackType(ScanSettings.CALLBACK_TYPE_ALL_MATCHES)
        .setMatchMode(ScanSettings.MATCH_MODE_AGGRESSIVE)
        .setNumOfMatches(ScanSettings.MATCH_NUM_ONE_ADVERTISEMENT)
        .setReportDelay(0L)
        .build();
```

Connect to a GATT Server

Connect to a Device

Call `connectGatt()`, returns a BluetoothGatt object that we'll later use for reading and writing

```
BluetoothGatt connectGatt(Context context,  boolean autoConnect,
BluetoothGattCallback callback,  int transport)
```

transport parameter:

- `TRANSPORT_LE`

autoconnect:

- false: **try to connect for 30 seconds**
- true: connect whenever Android sees the device, **never time out**

Connect State

`onConnectionStateChange` callback

```
public void onConnectionStateChange(final BluetoothGatt gatt, final int status,
final int newState)
```

newState

- STATE_CONNECTED
- STATE_DISCONNECTED
- STATE_CONNECTING
- STATE_DISCONNECTING

Status field is an error code

Transfer BLE data

Facts about Read and Write :

- Read/write operations are asynchronous
- One asynchronous operation at a time

Best Practice:

- Implement a command queue

```
private Queue<Runnable> commandQueue;
private boolean commandQueueBusy;
```

Read and Write

Command queue:

- Add a new Runnable to the queue when we do a command
- If successful added command to queue, start executing
    - peek() to obtain the Runnable from the queue, Leave the runnable in the queue for retry later. Set `commandQueueBusy` to true.
    - Set `commandQueueBusy` to false after processing the value
    - Take the Runnable off the queue by calling poll() and start the next command

Writing characteristics

In order to write data, we must provide the characteristics

- `WRITE_TYPE_DEFAULT`
- `WRITE_TYPE_NO_RESPONSE`

```
characteristic.setValue(bytesToWrite);
characteristic.setWriteType(writeType);
if (!bluetoothGatt.writeCharacteristic(characteristic)) {
    Log.e(TAG, String.format("ERROR: writeCharacteristic failed for
characteristic: %s", characteristic.getUuid()));
    completedCommand();
} else {
    Log.d(TAG, String.format("writing <%s> to characteristic <%s>",
bytes2String(bytesToWrite), characteristic.getUuid()));
    nrTries++;
}
```

Turning notifications on/off

If notification is on, the device will let us know when there is new data and send it to you automatically.

To tuen on notification:

- Call setCharacteristicNotification
- Write value 1 or 2 as a unsigned in16 to the Client Characteristic Configuration.

Bonding

Encryption keys are being generated, exchanged and stored so that the communication can become more secure.

- **Let Android handle bonding**
- **Don’t do anything while bonding is in progress**

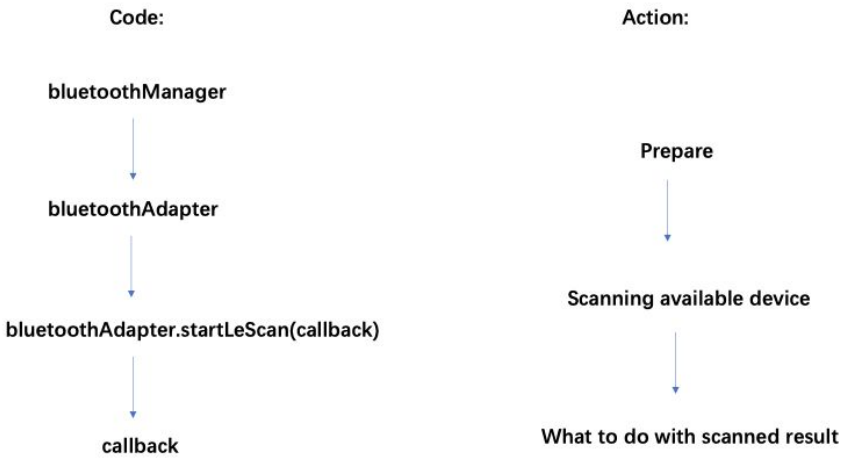
Losing a bond

- Phone A bonds with device X
- Phone B bonds with device X
- Phone A reconnects to device X, now the bond is **lost**

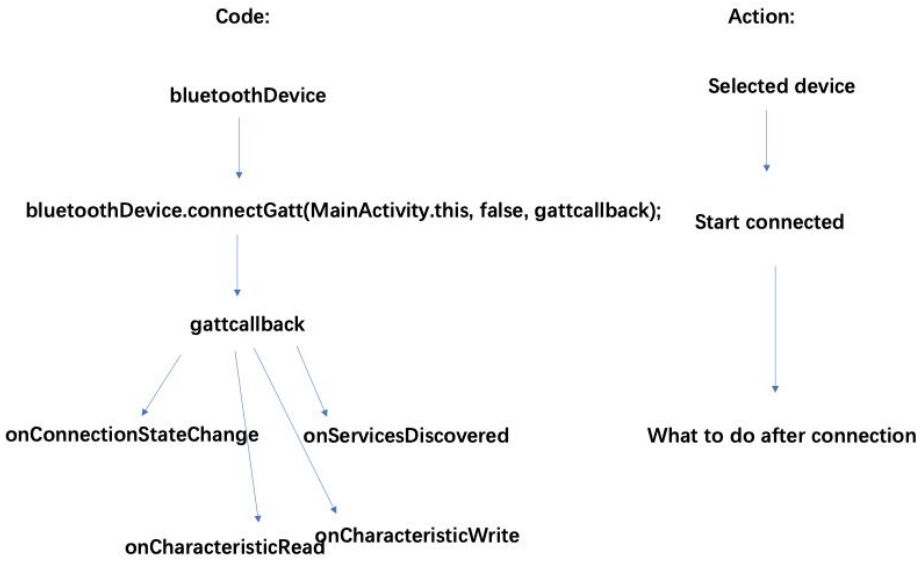
Disconnect

- Call `disconnect()`
- Wait for the `callback on onConnectionStateChange` to come in
- Call `close()`
- Dispose the gatt object

BLE Demo


Code:
bluetoothManager
bluetoothAdapter
bluetoothAdapter.startLeScan(callback)
callback
Action:
Prepare
Scanning available device
What to do with scanned result

Set up


Code:
Action:
bluetoothDevice
Selected device
bluetoothDevice.connectGatt(MainActivity.this, false, gattcallback);
Start connected
gattcallback
onConnectionStateChange
onServicesDiscovered
What to do after connection
onCharacteristicRead
onCharacteristicWrite

ref

Bluetooth Low Energy

What is BLE (Bluetooth Low Energy)? How does BLE work?

The Ultimate Guide to Android Bluetooth Low Energy

Bluetooth Technology Overview | Bluetooth® Technology Website

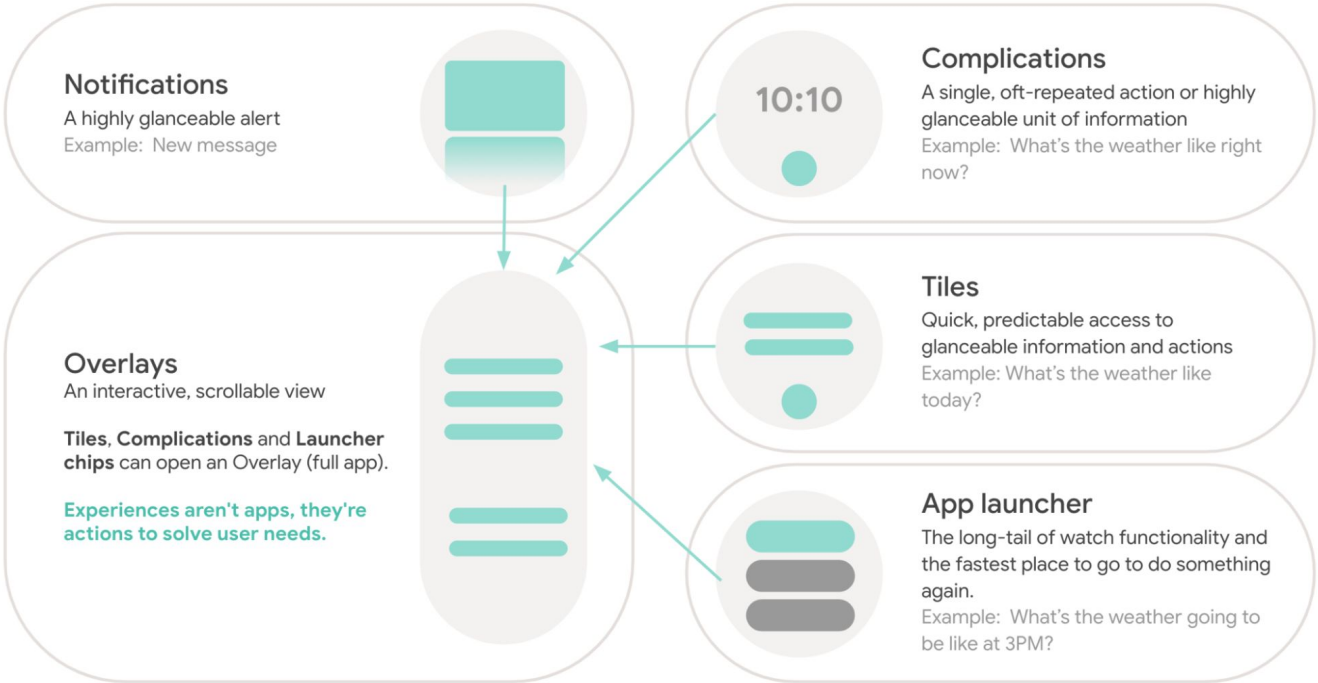
Bluetooth Low Energy (BLE) Beacon Technology Made Simple: A Complete Guide to Bluetooth Beacons

Making Android BLE work — part 1. BLE on Android is hard! Here is what I… | by Martijn van Welie

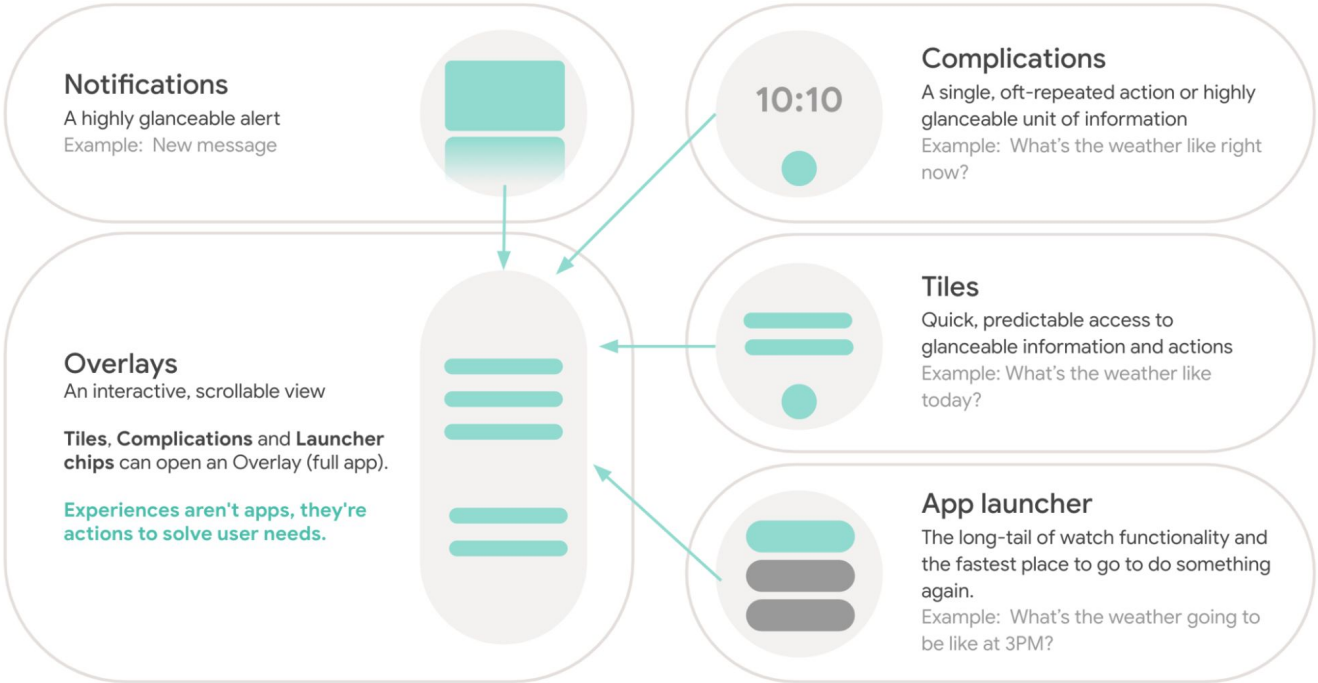
Wear UI

Zerong Li, Ching Po Lin, Di Zhu

Wear OS Overview



Wear OS Overview



Wear OS vs Mobile Development

UI Design	Focus on critical task
UI Components	BoxInsetLayout WearableRecyclerView SwipeDismisssFrameLayout
Theme	Wear OS always use Dark Theme to save battery
Navigation	Wear OS apps should have no more than two levels (screen and confirmation page) and linear.
Back	Use swipe to dismiss and control

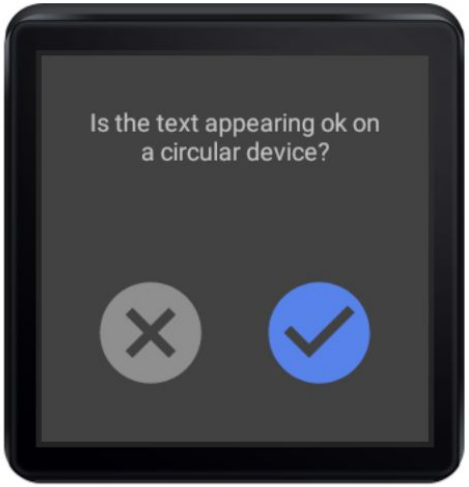
Handle different watch shapes

Use a BoxInsetLayout

Use different layouts for square and round screens

Use a curved layout (WearableRecyclerView)

Handle different watch shapes



The BoxInsetLayout class in the Wear OS UI Library lets you define a single layout that works for both square and round screens.

```
<androidx.wear.widget.BoxInsetLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:padding="15dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="5dp"
        app:layout_boxedEdges="all">

        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="@string/sometext"
            android:textAlignment="center"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
```

Handle different watch shapes

- [x] **Use a BoxInsetLayout**

Use different layouts for square and round screens

Use a curved layout (WearableRecyclerView)

Use different layouts for square and round screens

```
<dimen name="header_start_padding">36dp</dimen>
<dimen name="header_end_padding">22dp</dimen>
<dimen name="list_start_padding">36dp</dimen>
<dimen name="list_end_padding">22dp</dimen>
```

Figure 4. Using `values-round/dimens.xml`.

```
<dimen name="header_start_padding">16dp</dimen>
<dimen name="header_end_padding">16dp</dimen>
<dimen name="list_start_padding">10dp</dimen>
<dimen name="list_end_padding">10dp</dimen>
```

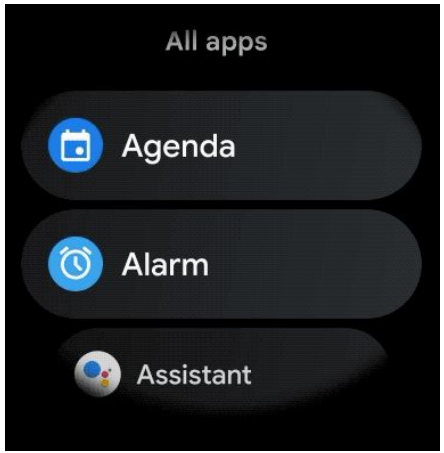
Figure 5. Using `values/dimens.xml`.

Handle different watch shapes

- [x] Use a BoxInsetLayout
- [x] Use different layouts for square and round screens
- [ ] Use a curved layout (WearableRecyclerView)

WearableRecyclerView

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.wear.widget.WearableRecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main_menu_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical"
    android:background="@color/white"/>
```



Handle different watch shapes

- Use a BoxInsetLayout
- Use different layouts for square and round screens
- Use a curved layout (WearableRecyclerView)

Wear OS vs Mobile Development

UI Design	Focus on critical task
UI Components	BoxInsetLayout WearableRecyclerView SwipeDismisssFrameLayout
Theme	Wear OS always use Dark Theme to save battery
Navigation	Wear OS apps should have no more than two levels (screen and confirmation page) and linear.
Back	Use swipe to dismiss and control

SwipeDismisssFrameLayout

```
<androidx.wear.widget.SwipeDismissFrameLayout
    android:id="@+id/map_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
  <fragment
      android:id="@+id/map"
      android:name="com.google.android.gms.maps.SupportMapFragment"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />
</androidx.wear.widget.SwipeDismissFrameLayout>
```

Ongoing Activity

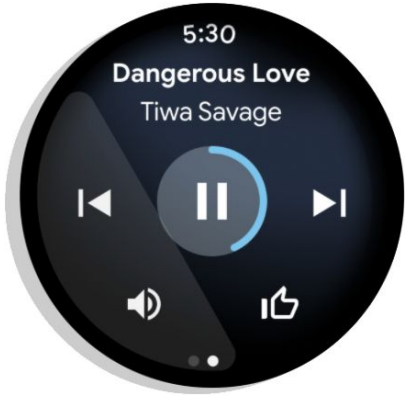
Pair an **ongoing activity** with an **ongoing notification**

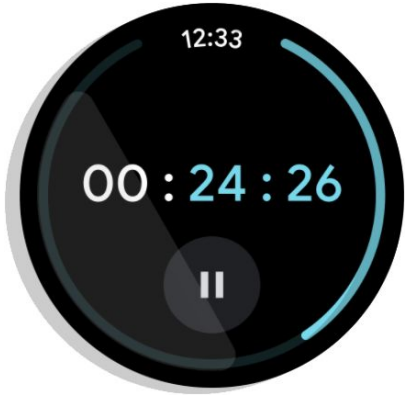
Ongoing Activity **API**

Expose information to multiple convenient surfaces on Wear OS

Keep the user engaged

Ongoing Activity Example





Setup

App's `build.gradle` file

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    implementation "androidx.wear:wear-ongoing:1.0.0"
    implementation "androidx.core:core-ktx:1.7.0"
```

Ongoing Notification

Closely related to Ongoing Notification

- The only surface your app will show
- As the communication mechanism

Status

Tiles

-- Easy access

-- Simple swipe

Overview of Building custom tiles

Creation:

-- Extend TileService and complete Override ListenableFuture (Tile/resources)

-- Adding services and permission in android manifest file inside application tag

-- Preview the Tile by wear-tiles-renderer library

Implementation:

-- Timelines .

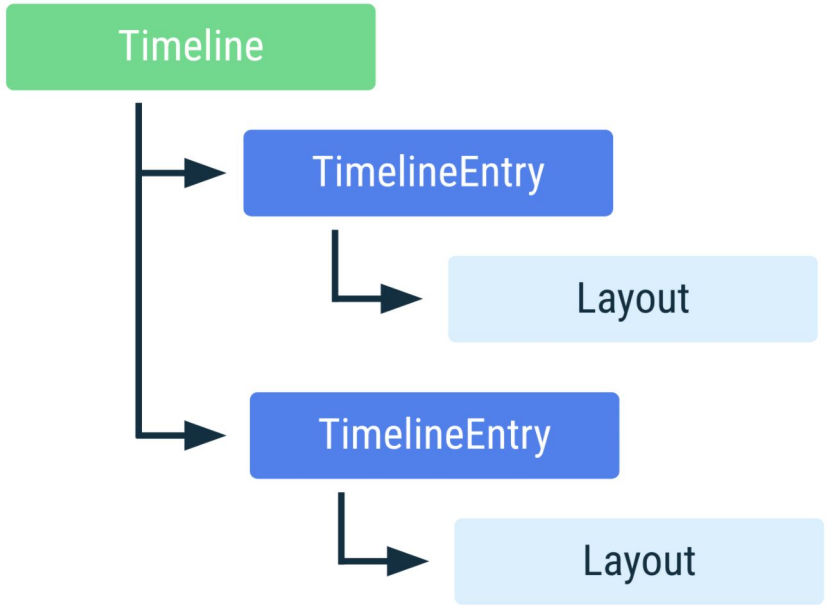
-- Refresh a Tile

-- Layout Design

-- Resources

-- Interaction: LaunchAction and LoadAction

Timeline



Building Layout

-- Tree shaped code

-- Using layout container

```
private LayoutElement myLayout() {
    return new Row.Builder()
        .setWidth(wrap())
        .setHeight(expand())
        .setVerticalAlignment(VALIGN_BOTTOM)
        .addContent(new Text.Builder()
            .setText("Hello world")
            .build()
        )
        .addContent(new Image.Builder()
            .setResourceId("image_id")
            .setWidth(dp(24f))
            .setHeight(dp(24f))
            .build()
        ).build();
}
```