Advanced Touch Input

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

Contents

Moving/Resizing a View

What?

- The ability to move and resize view at **runtime**

What?

- The ability to move and resize view at **runtime**
  - What's runtime?

What?

- The ability to move and resize view at **runtime**
  - What's runtime?
- Interface is modified Based on the user interaction

What?

- The ability to move and resize view at **runtime**
  - What's runtime?
- Interface is modified Based on the user interaction
- Interactive views

Why?

- Adaptive user interface

Why?

- Adaptive user interface
- Contribute to to better user experience

Why?

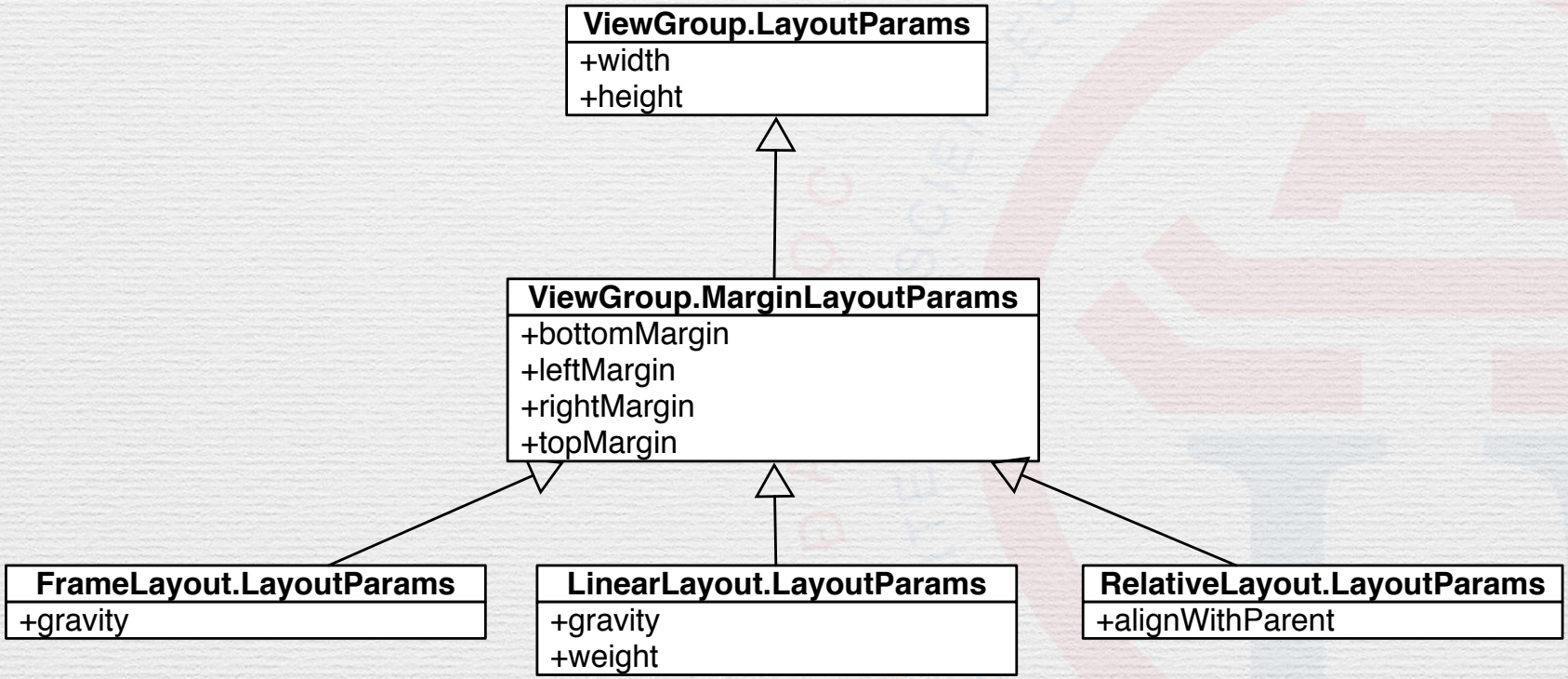
- Adaptive user interface
- Contribute to to better user experience
- Cool

How?

- LayoutParams
  - Layout Parameters
  - Different type for each parent view group
- To modify size and position of a View
  - Get its LayoutParam with the type **from its parent**
  - Modify position and/or size from the accquired LayoutParams
  - Set the LayoutParams back to the **View** (not the parent)
- You have to know the parent view's type before hand...

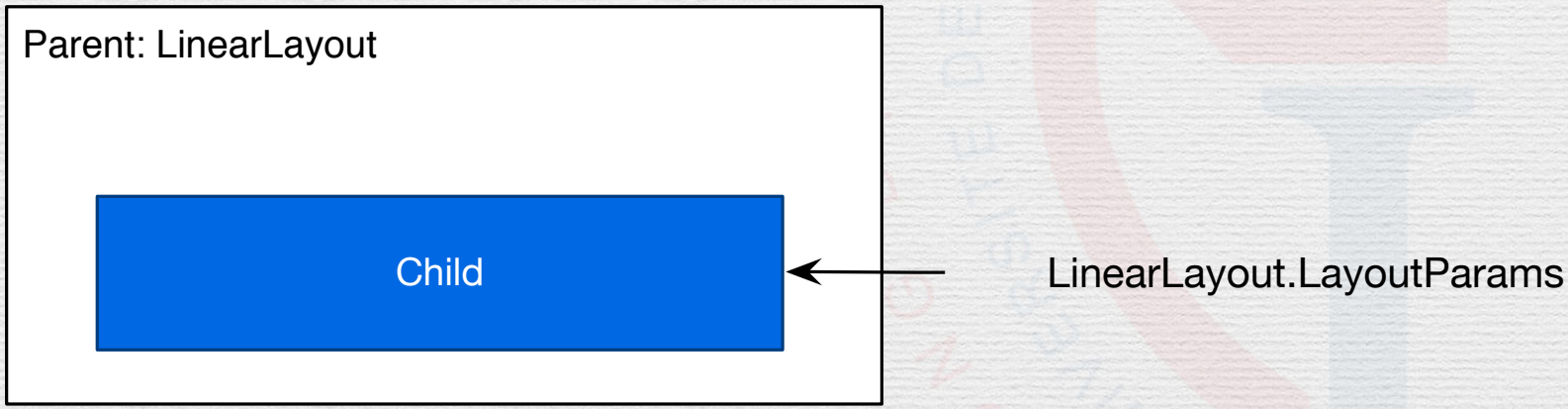
How?

- What's there in LayoutParams?



How?

- Type: parent layout params
  - `RelativeLayout.LayoutParams`
  - `LinearLayout.LayoutParams`
  - `RelativeLayout.LayoutParams`
- Get layout param of the child with the type from parent



How: Example

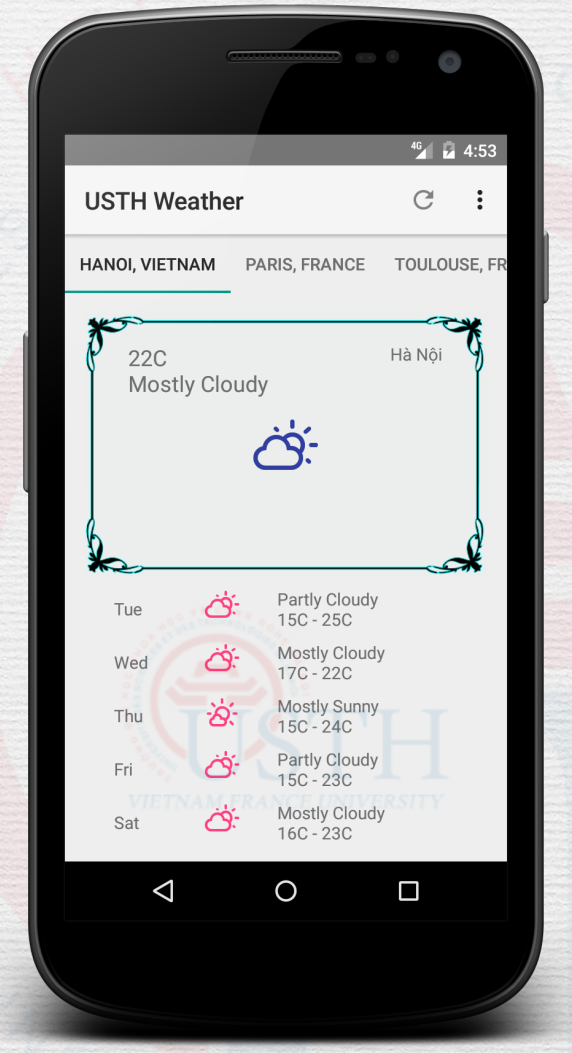
```
// Get its LayoutParam with the type from its parent
LinearLayout.LayoutParams lp =
    (LinearLayout.LayoutParams) child.getLayoutParams();

// Modify position and/or size from the accquired LayoutParams
lp.width /= 2;
lp.height /= 2;

// Set the LayoutParams back to the View (not the parent)
child.setLayoutParams(lp);
```

Practical Work 18

- Half the size of weather icon in WeatherFragment everytime it's clicked
- Use `child.getWidth()` instead of lp.width if you use `WRAP_CONTENT` or `MATCH_PARENT`



Simple single-touch gestures

What?

- Hold

What?

- Hold
- Drag

What?

- Hold
- Drag
- Fling

Why?

- Standard user experience
- Basic interaction with user
- Improve interaction for non-standard views

How?

- Two main ways:
  - Manual analyzer with `onTouch`
  - «Simplified» with `GestureDetector`

How: Manual Analyzer

- Use `View.setOnTouchListener`

How: Manual Analyzer

- Use `View.setOnTouchListener`
- Override `onTouch`

How: Manual Analyzer

- Use `View.setOnTouchListener`
- Override `onTouch`
- Analyze touch information in `onTouch`

How: Manual Analyzer

- Use `View.setOnTouchListener`
- Override `onTouch`
- Analyze touch information in `onTouch`
  - Type: `MotionEvent.getActionMasked()`

How: Manual Analyzer

- Use `View.setOnTouchListener`
- Override `onTouch`
- Analyze touch information in `onTouch`
  - Type: `MotionEvent.getActionMasked()`
    - `ACTION_DOWN`: finger starts touching view
    - `ACTION_MOVE`: finger moves (still touching the screen)
    - `ACTION_UP`: finger lifts from view

How: Manual Analyzer

- Use `View.setOnTouchListener`
- Override `onTouch`
- Analyze touch information in `onTouch`
  - Type: `MotionEvent.getActionMasked()`
    - `ACTION_DOWN`: finger starts touching view
    - `ACTION_MOVE`: finger moves (still touching the screen)
    - `ACTION_UP`: finger lifts from view
  - Find touch position:

How: Manual Analyzer

- Use `View.setOnTouchListener`
- Override `onTouch`
- Analyze touch information in `onTouch`
  - Type: `MotionEvent.getActionMasked()`
    - `ACTION_DOWN`: finger starts touching view
    - `ACTION_MOVE`: finger moves (still touching the screen)
    - `ACTION_UP`: finger lifts from view
  - Find touch position:
    - `MotionEvent.getX()`: relative x-axis position (to view)
    - `MotionEvent.getY()`: relative y-axis position (to view)

How: Manual Analyzer

- Use `View.setOnTouchListener`
- Override `onTouch`
- Analyze touch information in `onTouch`
  - Type: `MotionEvent.getActionMasked()`
    - `ACTION_DOWN`: finger starts touching view
    - `ACTION_MOVE`: finger moves (still touching the screen)
    - `ACTION_UP`: finger lifts from view
  - Find touch position:
    - `MotionEvent.getX()`: relative x-axis position (to view)
    - `MotionEvent.getY()`: relative y-axis position (to view)
    - `MotionEvent.getRawX()`: absolute x-axis position (on screen)

How: Manual Analyzer

```
public class TouchActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        setContentView(R.layout.activity_main);

        // we apply our touch gestures only for the logo
        View v = findViewById(R.id.logo);
        v.setOnTouchListener(
            new View.OnTouchListener() {
                @Override
                public boolean onTouch(View view, MotionEvent motionEve
                    checkTouch(v, motionEvent);     // see next slide
                    return false;
                }
            });
    }

    // to be continued...
```

How: Manual Analyzer

```
// ... continuing from previous slide.
/**
 * Classify a touch input event.
 */
private void checkTouch(View v, MotionEvent motionEvent) {
    int action = event.getActionMasked();
    int posX = (int) event.getRawX(0);
    int posY = (int) event.getRawY(0);
    switch (action) {
        case MotionEvent.ACTION_DOWN:   // that's a touch start
            startTouch(v, event, posX, posY);
            break;
        case MotionEvent.ACTION_MOVE:   // finger is moving
            updateTouch(v, event, posX, posY);
            break;
        case MotionEvent.ACTION_UP:     // finished touch
            finishTouch(v, event, posX, posY);
            break;
    }
}
```

How: Manual Analyzer

```
// ... still continuing from previous slide.
int startTouchX, startTouchY;
int startViewX, startViewY;

/**
 * Process start touch event: save initial X and Y position
 */
private void startTouch(View v, MotionEvent motionEvent,
    int posX, int posY) {
    // save initial X and Y positions so that we can know
    // how much we should move the view later, in updateTouch()
    startTouchX = posX;
    startTouchY = posY;

    FrameLayout.LayoutParams lp =
        (FrameLayout.LayoutParams) v.getLayoutParams();
    startViewX = lp.leftMargin;
    startViewY = lp.topMargin;
}
// to be continued
```

How: Manual Analyzer

```
// ... still continuing from previous slide.

/**
 * Process touch move event
 */
private void updateTouch(View v, MotionEvent motionEvent,
    int posX, int posY) {

    // move the view according to the current touch position
    int dx = posX - startTouchX;
    int dy = posY - startTouchY;

    FrameLayout.LayoutParams lp =
        (FrameLayout.LayoutParams) v.getLayoutParams();
    lp.leftMargin = startViewX + dx;
    lp.topMargin = startViewY + dx;
    v.setLayoutParams(lp);
}
// to be continued...
```

How: Manual Analyzer

```
        // ... still continuing from previous slide.

        /**
         * Process touch lifted event
         */
        private void finishTouch(View v, MotionEvent motionEvent,
            int posX, int posY) {
            // basically we only need to reset the position.
            startViewX = 0;
            startViewY = 0;
            startTouchX = 0;
            startTouchY = 0;
        }
    }

    // that's it. I know you're already overwhelmed ;)
```

Practical Work: 19

- Time for copy - paste...
- Create a new activity, no fragment
- Root layout: FrameLayout
  - `MATCH_PARENT` / `MATCH_PARENT`
- Add an ImageView inside a FrameLayout
  - id: logo
  - src: use USTH Logo
- Implement onTouchListener from previous slides
  - Move Logo with touches

How: GestureDetector

- What?

How: GestureDetector

- What? A class for detecting common gestures

How: GestureDetector

- What? A class for detecting common gestures
- Why?

How: GestureDetector

- What? A class for detecting common gestures
- Why?
  - Don't reinvent the wheel...

How: GestureDetector

- What? A class for detecting common gestures
- Why?
  - Don't reinvent the wheel... and...

How: GestureDetector

- What? A class for detecting common gestures
- Why?
  - Don't reinvent the wheel... and...
  - You guys are all lazy...

How: GestureDetector

1. Create new class
   - Extends `GestureDetector.SimpleOnGestureListener`

How: GestureDetector

1. Create new class
   - Extends `GestureDetector.SimpleOnGestureListener`
2. Override desired methods, do what you want
   - `onDown()`
   - `onLongPress()`
   - `onFling()`
   - `onSingleTap()`
   - `onDoubleTap()`
   - `onScroll()`

How: GestureDetector

1. Create new class
   - Extends `GestureDetector.SimpleOnGestureListener`
2. Override desired methods, do what you want
   - `onDown()`
   - `onLongPress()`
   - `onFling()`
   - `onSingleTap()`
   - `onDoubleTap()`
   - `onScroll()`
3. Apply gesture detector in view's `setOnTouchListener`
   - Using GestureDetectorCompat

How: GestureDetector

1. Add new class inside (or outside) the activity

```
public class TouchActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) { ... }

    private class TouchGestureListener extends
        GestureDetector.SimpleOnGestureListener {

        ...
        // more on next slide
    }
}
```

How: GestureDetector

2. Override desired methods

```
// continue implementation of the TouchGestureListener
private class TouchGestureListener extends
    GestureDetector.SimpleOnGestureListener {

    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2,
        float distanceX, float distanceY) {
        int dx = (int) (e2.getX() - e1.getX());
        int dy = (int) (e2.getY() - e1.getY());
        FrameLayout.LayoutParams lp =
                (FrameLayout.LayoutParams) logo.getLayoutParams();
        lp.leftMargin = lp.leftMargin + dx;
        lp.topMargin = lp.topMargin + dy;
        logo.setLayoutParams(lp);
        return true;
    }
}
```

How: GestureDetector

3. Apply gesture detector in view's `setOnTouchListener`

```
public class TouchActivity extends AppCompatActivity {
    private View logo;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        logo = findViewById(R.id.logo);
        final GestureDetectorCompat detector =
          new GestureDetectorCompat(this, new TouchGestureListener());
        logo.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent motionEvent){
                return detector.onTouchEvent(motionEvent);
            }
        });
    }
}
```

Practical Work 20

- Improve your previous touch listener
  - Use `GestureDetector`
  - Move Logo with touches
  - If possible, implement «throw»

Multi-touch gestures

What?

- Gestures with two or more fingers
- Pinch-zoom
- Rotate

Why?

- Even more interactive user interface
- Even better user experience

How?

- Manual `onTouchEvent`
  - `ACTION_POINTER_DOWN`
  - `ACTION_POINTER_UP`

How?

We are out of time... Google yourself ☺