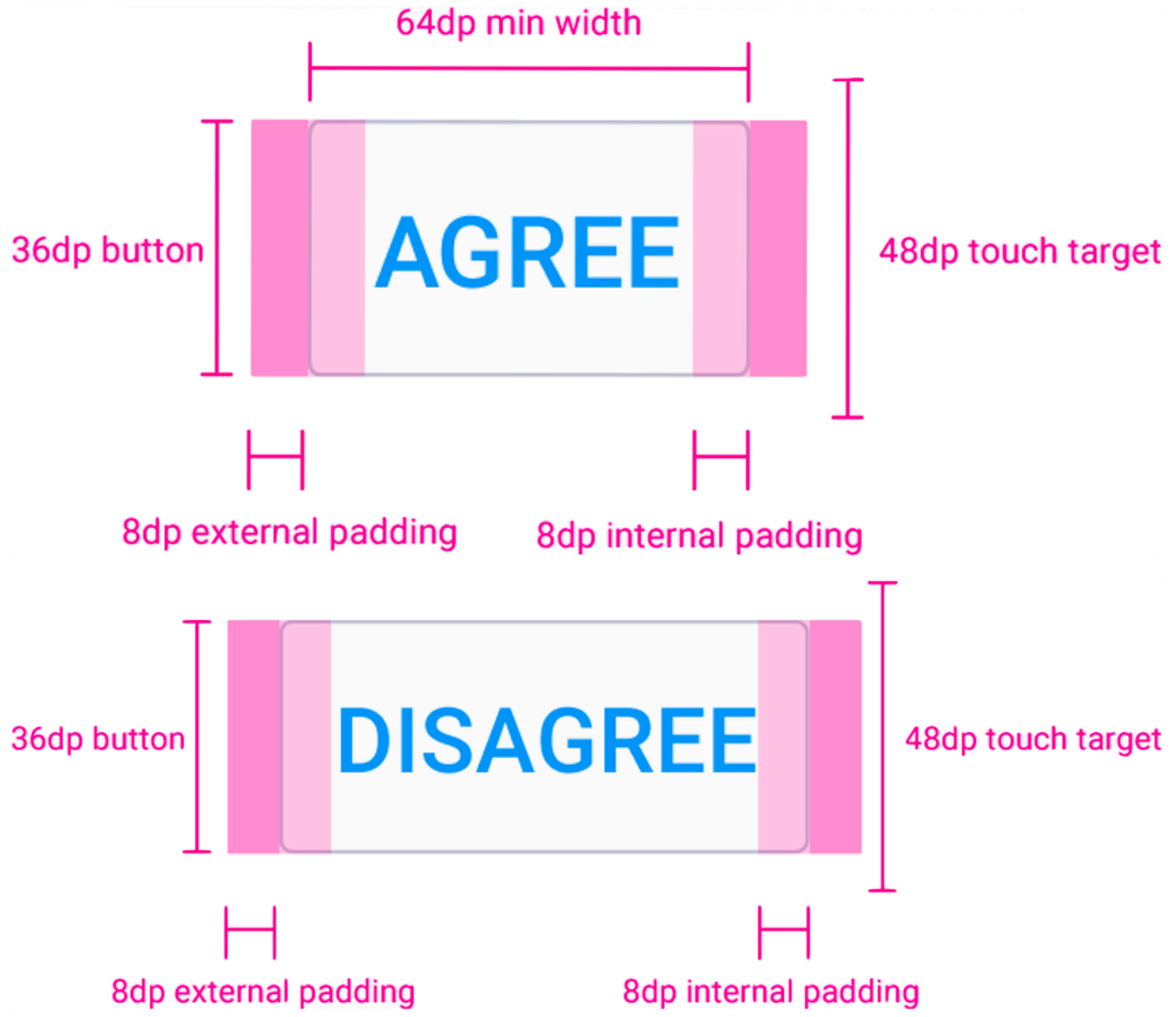


MOBILE APPLICATION DEVELOPMENT

ANDROID (2017)

LECTURE 07: MEASUREMENT

VIEW MEASUREMENT

- Android layout is at least a two-pass process.
  - First, the system does measure passes that determines the sizing of Views.
  - Second, the system does a layout pass in which Views are actually laid out.
- During the measure passes, Views are given restrictions and asked to calculate their preferred size within the specified restrictions.
- Parent Views send restrictions to child Views, which may modify the restrictions further before sending them to their child Views.
- At the end of the measure passes, all sizings meet the system's requirements.

MEASURESPECS AND LAYOUT PASSES

- The View.MeasureSpec class is the class Android uses to communicate restrictions on View measurements to Views during measure passes.
- This class provides size and mode pairs for a given dimension.
    - Size is specified either as an exact number, `MATCH_PARENT`, or `WRAP_CONTENT`.
    - Modes are either `UNSPECIFIED`, `EXACTLY`, or `AT_MOST`.
- Layout passes are initiated when either the programmer or the system calls `requestLayout()`. This triggers a full measure/layout pass of a View tree.
- Each View in the View tree has its `onMeasure()` function called during layout.

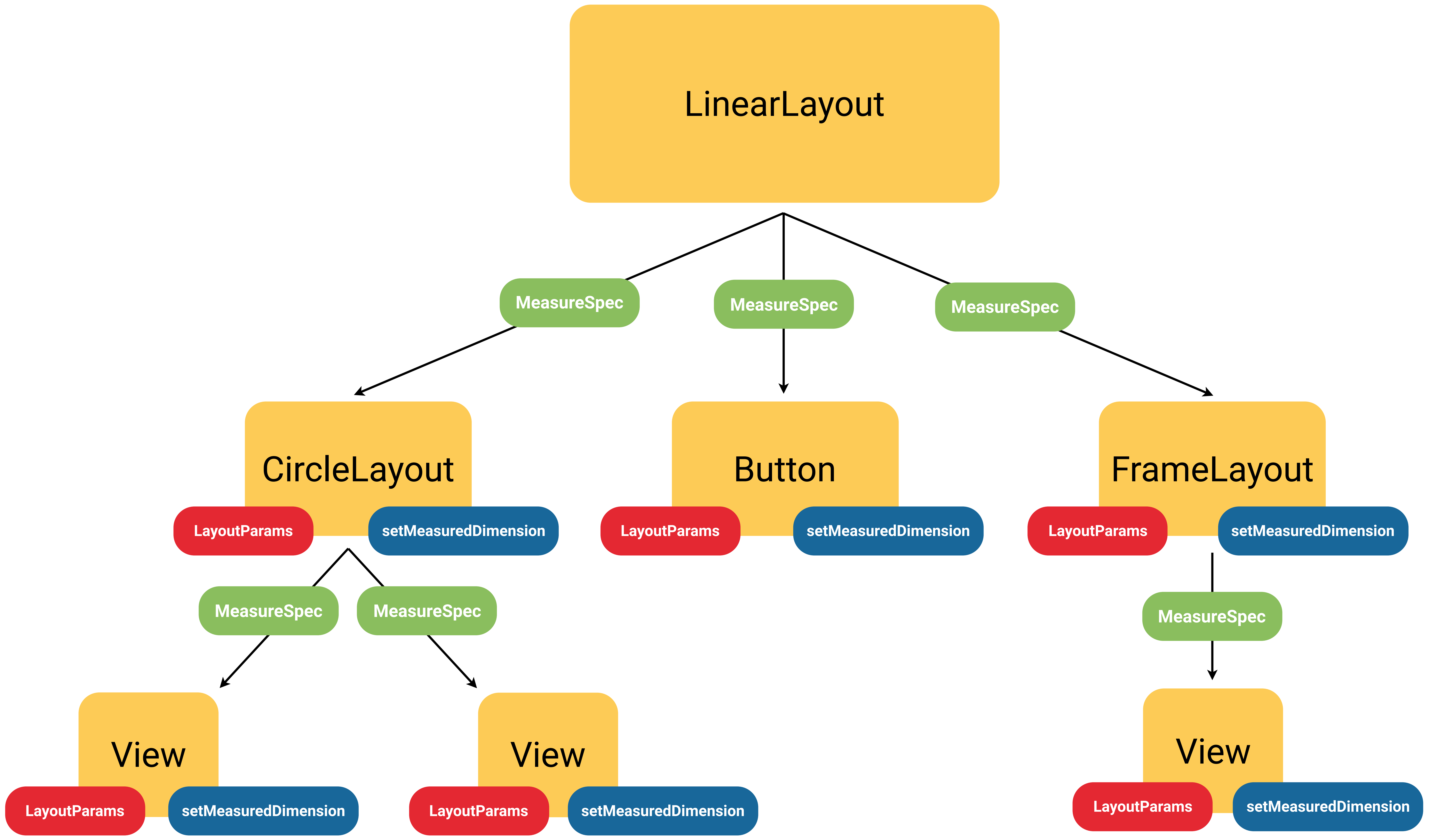
ON MEASURE FUNCTION

- The `onMeasure()` function is called in order to have a View specify the size it wants to be during layout. It is passed two View.MeasureSpec objects, one for the height of the View and one for the width.
- The function has the following objectives:
  - Must call `setMeasuredDimension()`.
  - Must ensure measured dimensions are at least the View's suggested minimums (obtained via `suggestedMinimumHeight` / `suggestedMinimumWidth`).
  - Should respect the dimensions in the View.MeasureSpec parameters by calling the `resolveSize()` function on each dimension before setting it.

WHEN TO USE MANUAL MEASUREMENT

- Overriding `onMeasure()` is not necessary, but is a good idea for custom Views.
- If `onMeasure()` is not overridden, Views may be incorrectly displayed.
    - If a View is told to use `wrap_content` for its layout width and height and doesn't have an implementation of `onMeasure()`, it will not take up any space.
    - If a View is supposed to be bigger than the default size Android uses, its content will get clipped.
    - If a View is supposed to be smaller than the default size Android uses, it will waste space and its content could appear off-center or stretched.

VISUALIZING MEASURE PASSES



MEASUREMENT CONSIDERATIONS

- Measurement may be more than a one-pass call.
  - Layouts may need to try measurement more than once to get sizes that work.
  - Measurement should be performant, since it might get called many times.
- Implementations of `onMeasure()` need to respond in a reasonable way.
  - If Views do not provide reasonable measurements, they may be forced into using measurement sizes they did not expect.
  - If Views over- or underestimate their measurements, it may cause other Views' `onMeasure()` to be called more times than necessary.