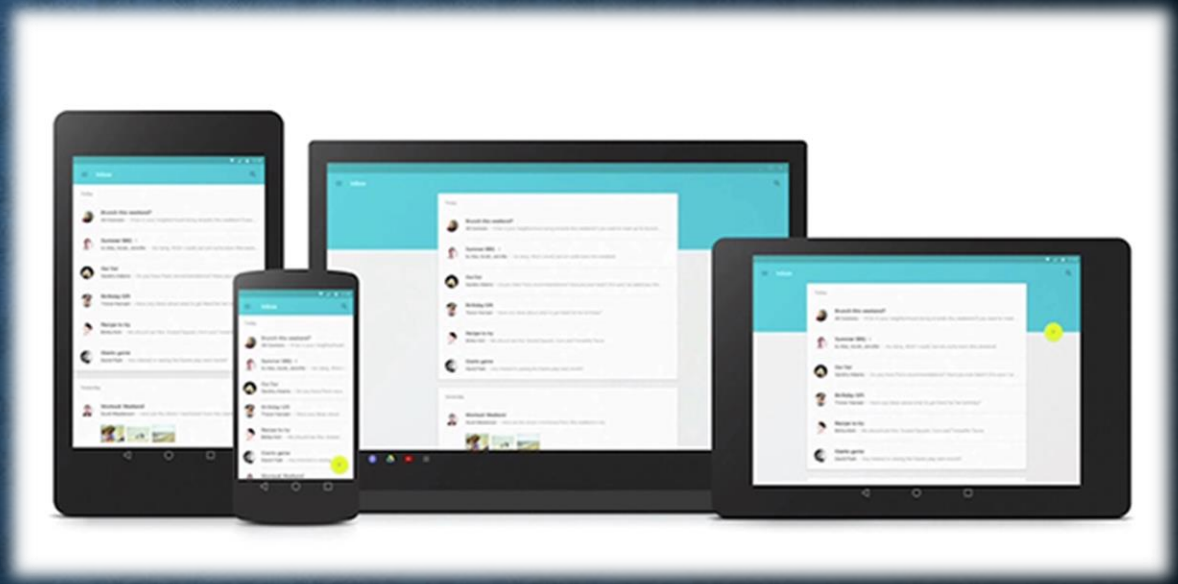
ANDROID

Material and Flexible Design

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

MATERIAL DESIGN - OVERVIEW

- A simple and flat UI design
- Keep it simple
- Responsive design for different devices/screens
- Designed with natural layout and quick navigation



MATERIAL DESIGN - GOALS

Create a visual language that synthesizes classic principles of good design with the innovation and possibility of technology and science

Develop a single underlying system that allows for a unified experience across platforms and device sizes

MATERIAL DESIGN - ENVIRONMENT

Material design is a three-dimensional environment containing

Light

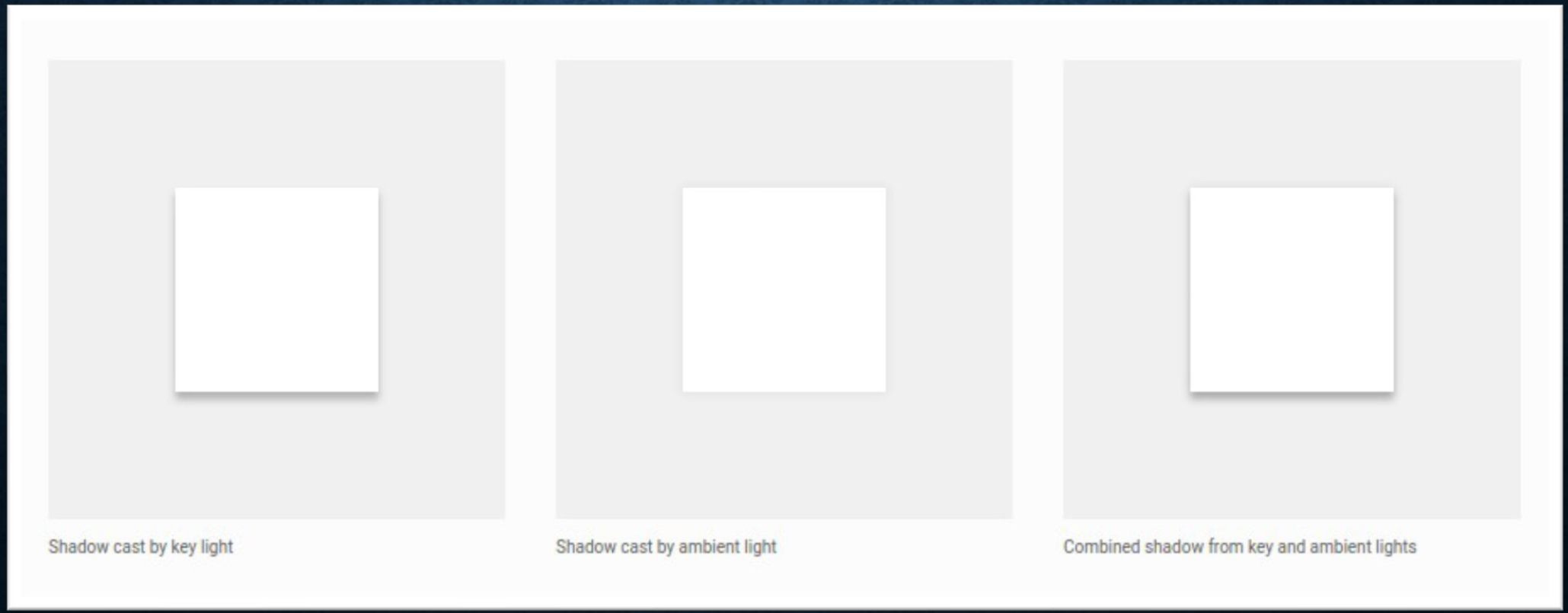
Material

Cast shadows

All objects have x, y, and z dimensions

Shadows are created by manipulating the y-axis

MATERIAL DESIGN - ENVIRONMENT



MATERIAL DESIGN - MOTION

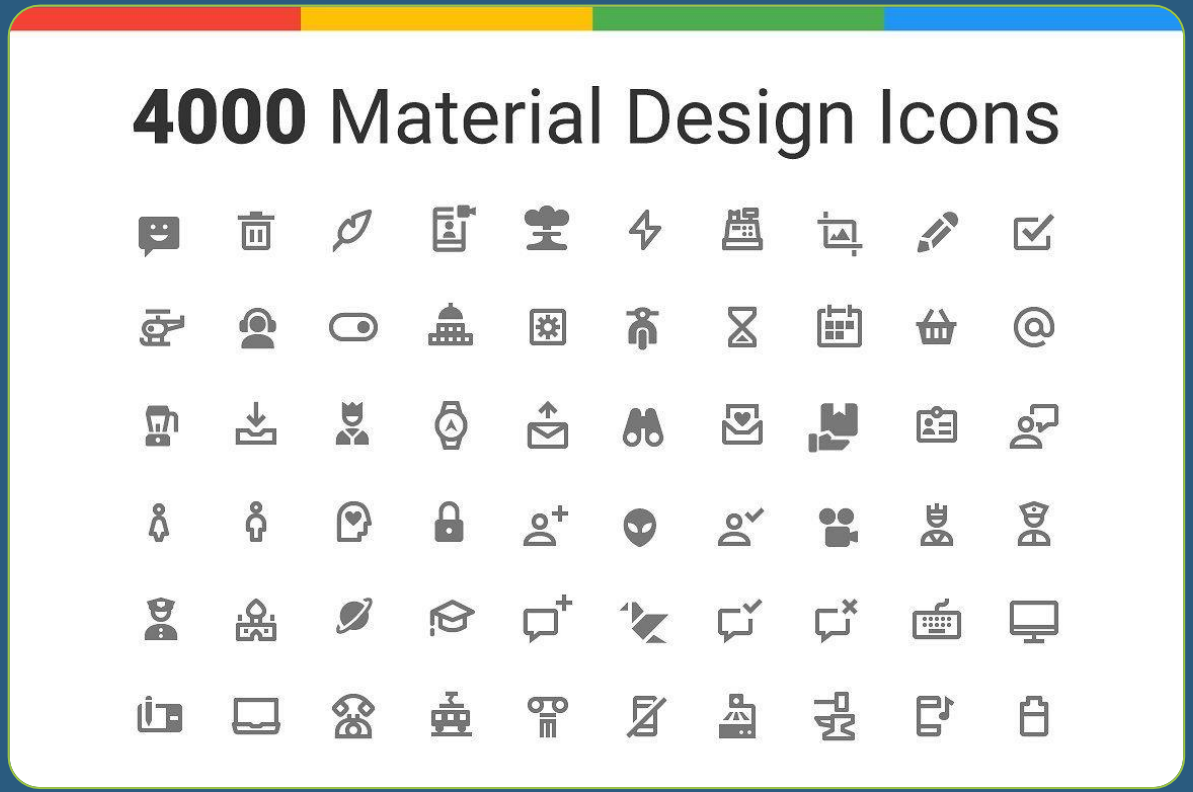
- Guided focus between views
- Hints at what will happen if a user completes a gesture
- Hierarchical and spatial relationships between elements
- Distraction from what is happening behind the scenes
  - Example: fetching content or loading the next view

MATERIAL DESIGN - COLORS

- Inspired by Bold hues
- Muted environments
- Deep Shadows
- Bright Highlights
- Color tool
  - https://material.io/color/#!/
  - Used to help you create a color theme that works

MATERIAL DESIGN - ICONS

- Product icons are the visual expression of a brand's products, services, and tools
- Product icon design is inspired by the tactile and physical quality of material
  - Each icon is cut, folded, and lit as paper would be, but represented by simple graphic elements
- System icons represent a command, file, device, directory, or common actions
  - System icons are also used to represent common actions like trash, print, and save



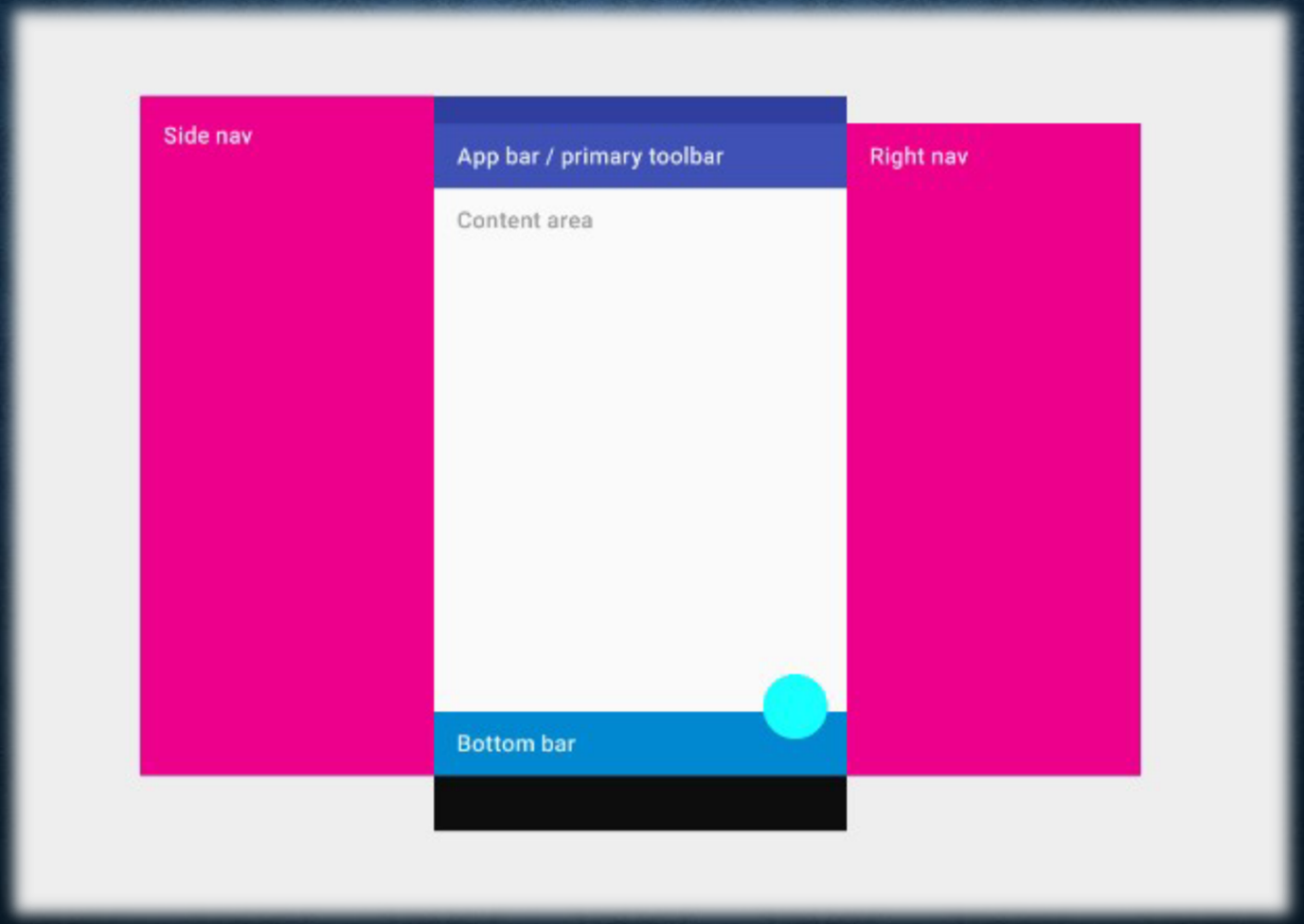
MATERIAL DESIGN - LAYOUT

- In material design, the physical properties of paper are translated to the screen
- The background of an application resembles the flat, opaque texture of a sheet of paper
- The app's behaviour mimics paper's ability to
  - Be re-sized
  - Shuffled
  - Bound together in multiple sheets

MATERIAL DESIGN - STRUCTURE

- Mobile structure
  - Permanent app bar and floating action button
  - Optional Bottom bar for additional functionality
  - Side navigation menus overlay all other elements

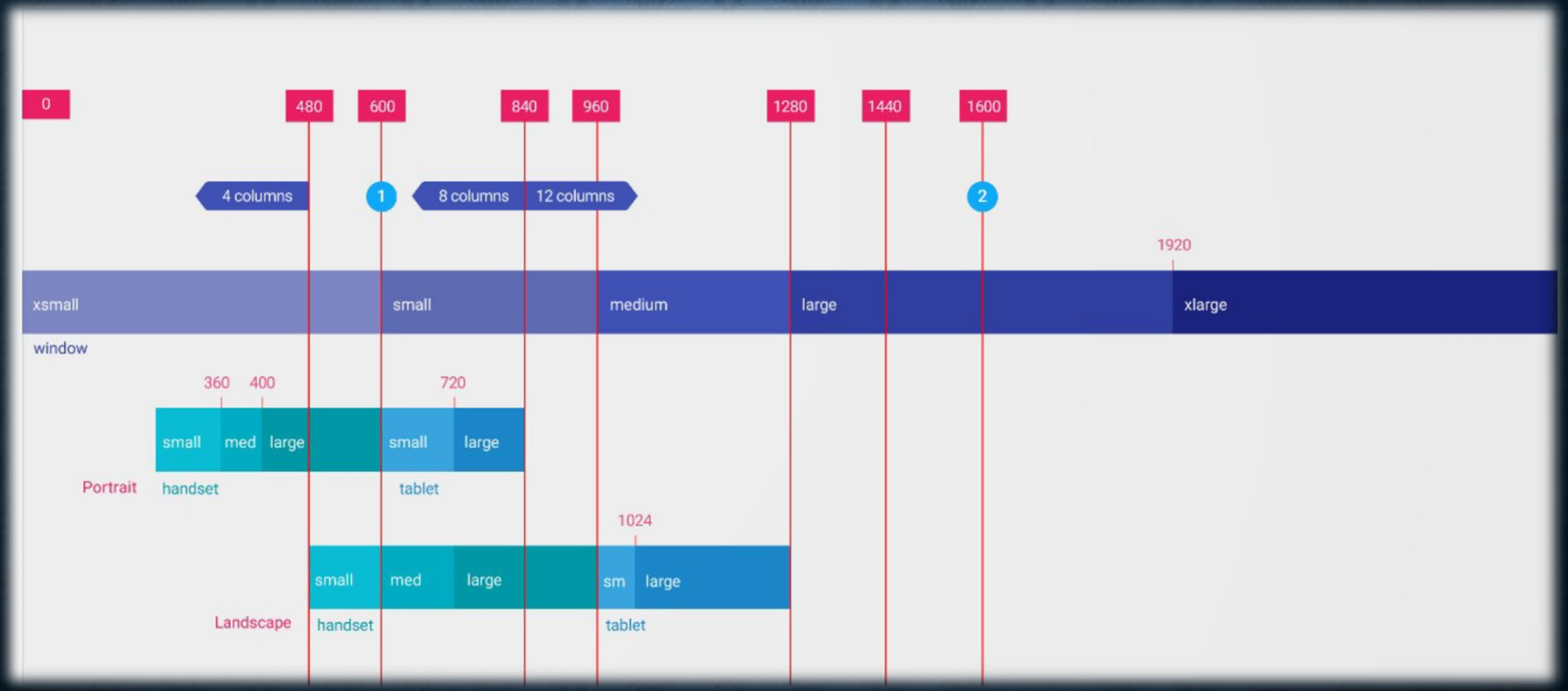
MATERIAL DESIGN - STRUCTURE



MATERIAL DESIGN - RESPONSIVE

- Responsive layouts in material design adapt to any possible screen size
- This UI guidance includes a flexible grid that ensures consistency across layouts
- A description of how an app can scale
  - From small screens (smart watches)
  - To extra-large screens (TVs)

MATERIAL DESIGN - RESPONSIVE

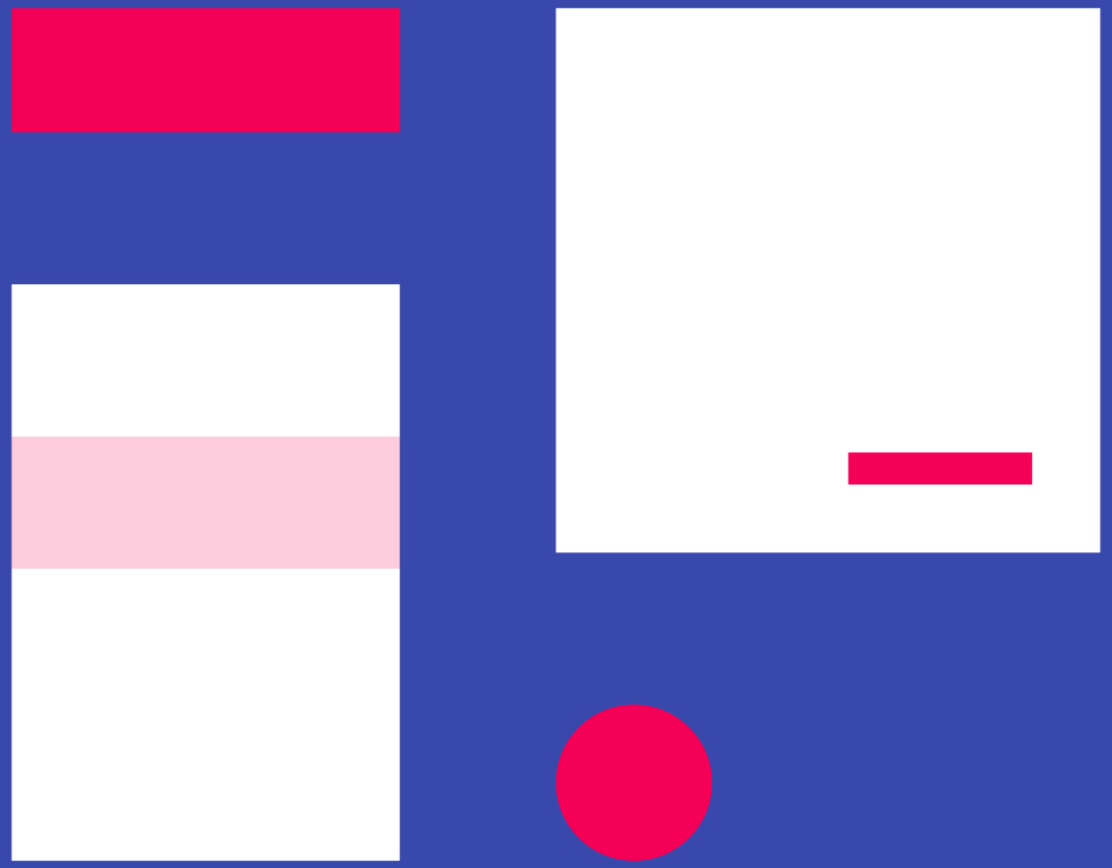


MATERIAL DESIGN - COMPONENTS

Bottom Navigation

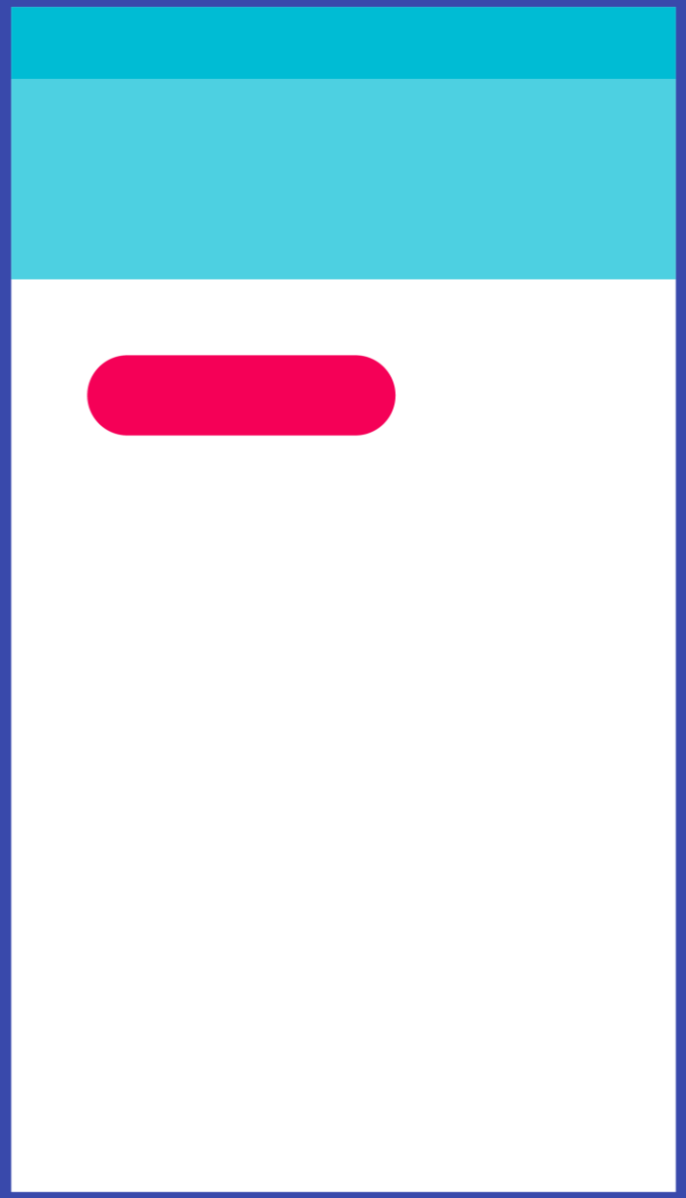
MATERIAL DESIGN - COMPONENTS

Buttons



MATERIAL DESIGN - COMPONENTS

Chips

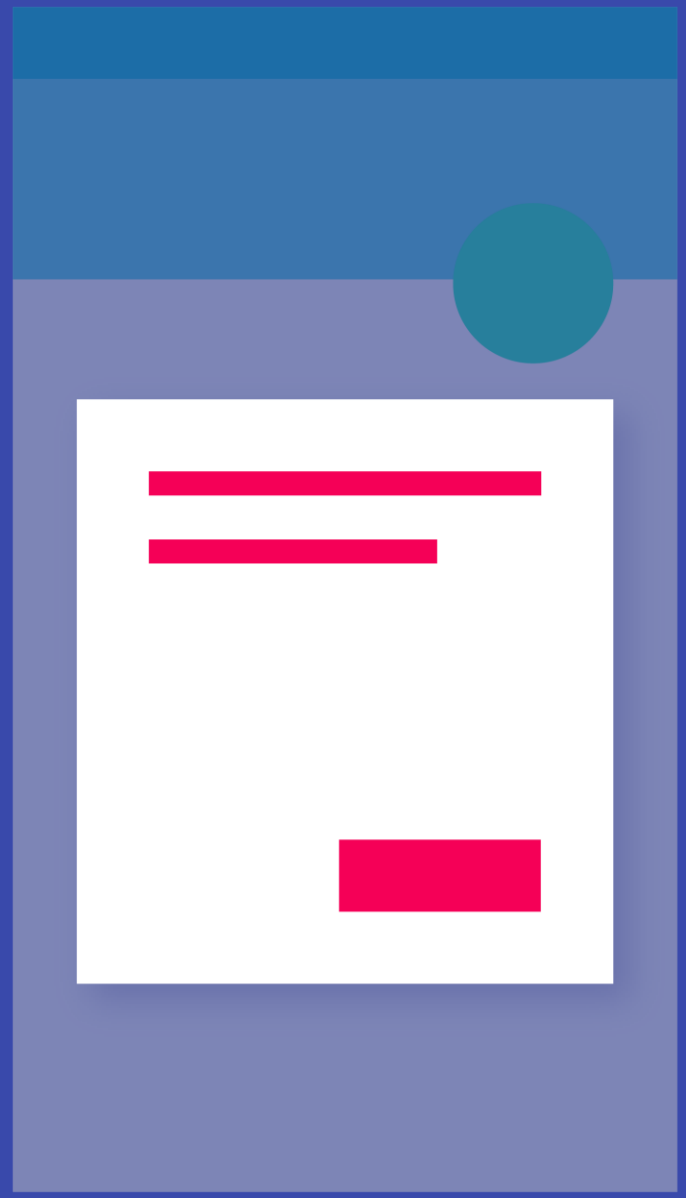


MATERIAL DESIGN - COMPONENTS

Cards

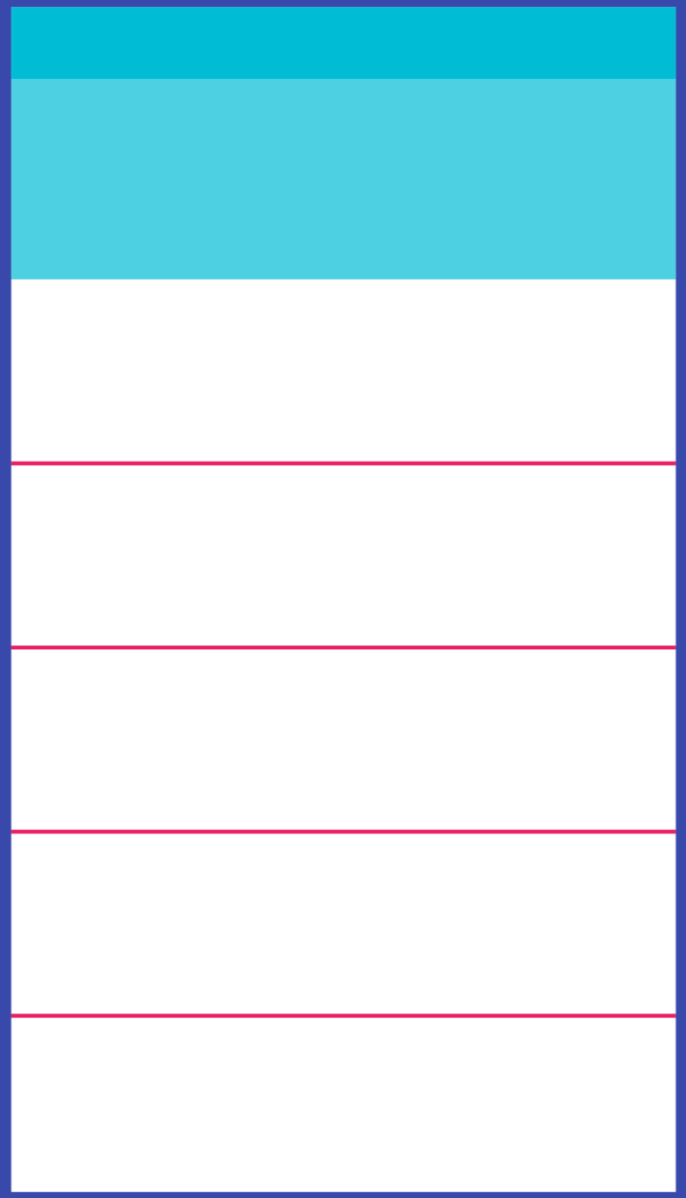
MATERIAL DESIGN - COMPONENTS

Dialogs



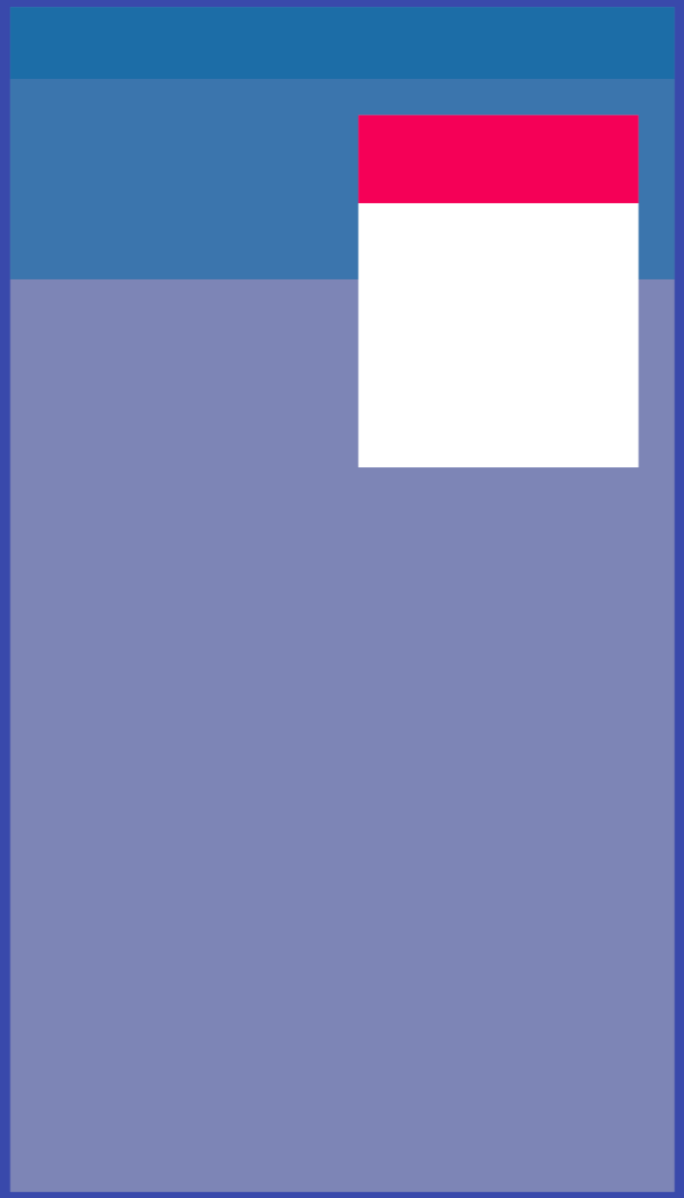
MATERIAL DESIGN - COMPONENTS

Dividers



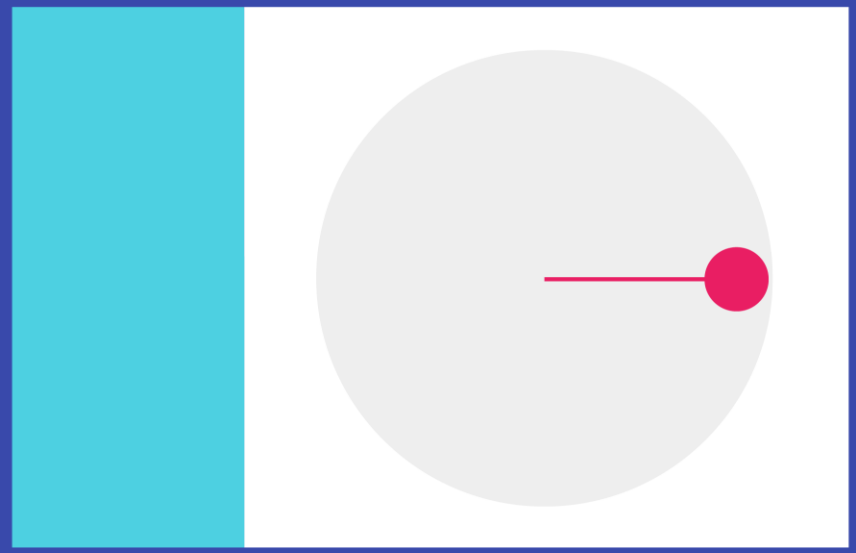
MATERIAL DESIGN - COMPONENTS

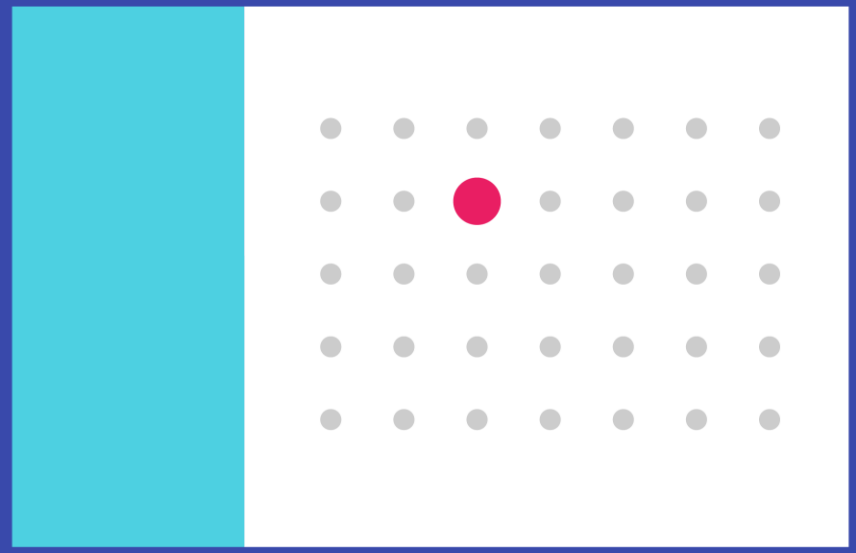
Menus



MATERIAL DESIGN - COMPONENTS

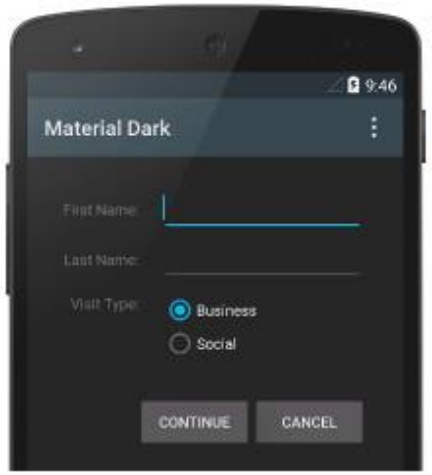
Pickers (Date and Time)



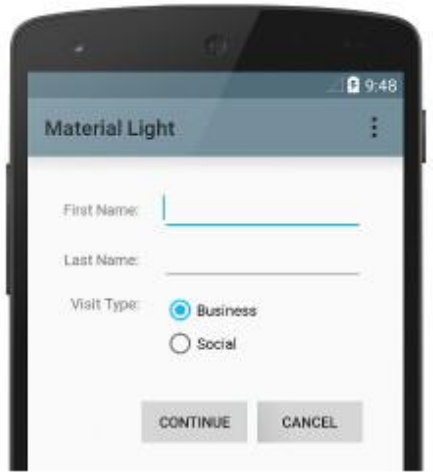


MATERIAL DESIGN - THEMES

- Various themes for material design
- Main two themes are the light and dark themes



Dark material theme



Light material theme

MATERIAL DESIGN - ANDROID

- Android has added a lot to the material design
  - Animations
  - View shadows
  - Integration into other Google products (Gmail, Google Maps, etc)

https://developer.android.com/design/material/index.html

RELATIVE SIZES

- Android supports many different screen sizes
- Instead of designing the layout with fixed sizes (e.g. 200px width), try to use relative sizes
  - wrap_content: uses the minimum width and height to fit the content of an element
  - match_parent: uses/fills the width and height of the parent element
- Use relative sizes for other parts too
  - Font size: ems (scale font sizes on different screens)
  - Padding, margin, etc: dp/dip (density independent pixel)

RELATIVE SIZES

```
<EditText
        android:id="@+id/text_message"
        android:inputType="textMultiLine"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:layout_marginRight="100dp"
/>
```

RELATIVE LAYOUT

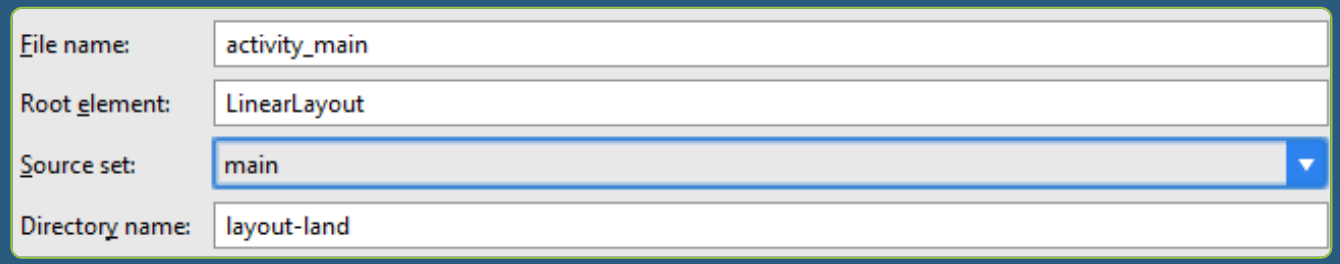
- Layout or align your views relative to each other
- Example: align on child view to the right of the parent, and another one to the left of the parent

RELATIVE LAYOUT

```
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        tools:context="com.statoshi.app.MainFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <EditText
                android:id="@+id/text_message"
                android:inputType="textMultiLine"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:ems="10"
                android:layout_marginRight="100dp" />
        <Button
                android:id="@+id/submit_button"
                android:inputType="textMultiLine"
                android:layout_width="100dp"
                android:layout_height="wrap_content"
                android:text="Submit"
                android:layout_alignParentTop="true"
                android:layout_alignParentRight="true" />
</RelativeLayout>
```

RELATIVE LAYOUT

- In order to use layouts for different screen orientations
  - Create a new resource file
  - Set its directory to "layout-land"
  - Give it the same name as the layout you want it to be of



SCREEN DENSITIES

- A pixel does not have a predefined size
- Depending on the device's screen resolution, fewer/more pixels might be available, and they might be of different size
- Do not use fixed pixel size, rather use
  - dp: density-independent pixels that corresponds to the physical size of a pixel at 160dpi
  - sp: same base unit as dp, but scaled by the user's preferred text size, only use for text, not layouts

FLEXIBLE UI

- When designing apps, support multiple screen sizes
- Use flexible design to rearrange views/layouts for
  - Different screen sizes
  - Different screen orientations (portrait and landscape)
- Fragments are perfect for flexible design

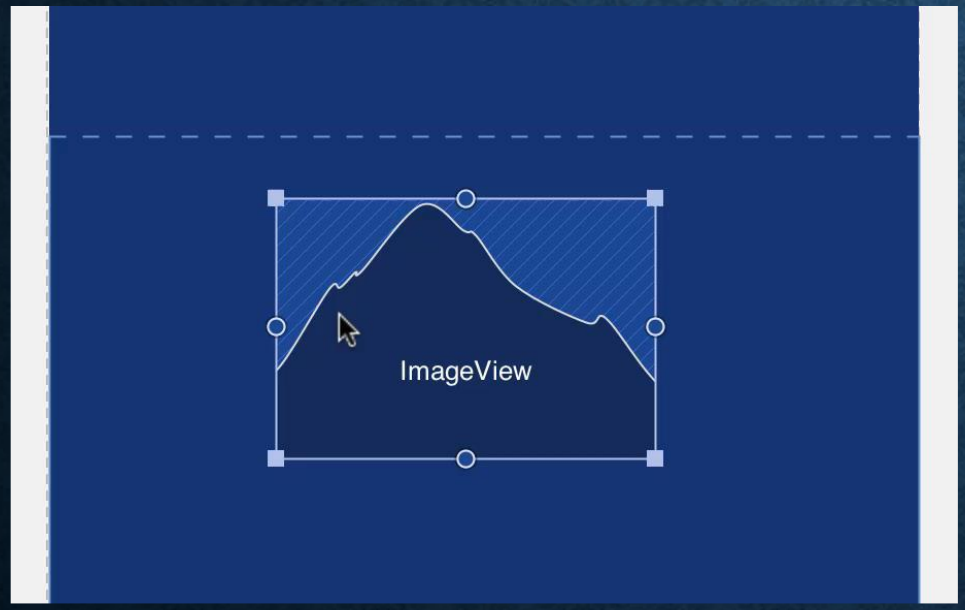
CONSTRAINT LAYOUT

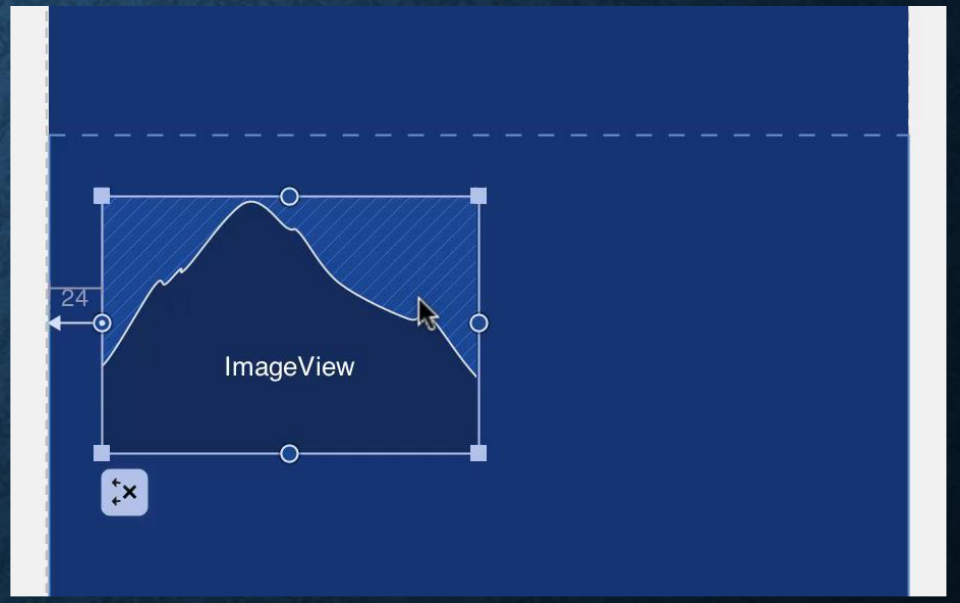
- ConstraintLayout allows you to create large and complex layouts with a flat hierarchy
  - No nested view groups
- ConstraintLayout is similar to RelativeLayout
  - Views are laid out according to relationships between sibling views and the parent layout
- ConstraintLayout is different to RelativeLayout
  - More flexible and responsive
  - Easier to use in Android Studio's Layout Editor

CONSTRAINT LAYOUT

- When creating constraints, keep the following in mind
  - Every view must have at least two constraints: one horizontal and one vertical
  - You can only create a constraint between a constraint handle and an anchor point that share the same plane. Hence, a vertical plane (left and right sides) of a view can be constraint only to another vertical plane; and baselines can only be constrained to another baseline
  - Each constraint handle can only be used for one constraint, but you can add multiple constraints (from different views) to the same anchor point

CONSTRAINT LAYOUT



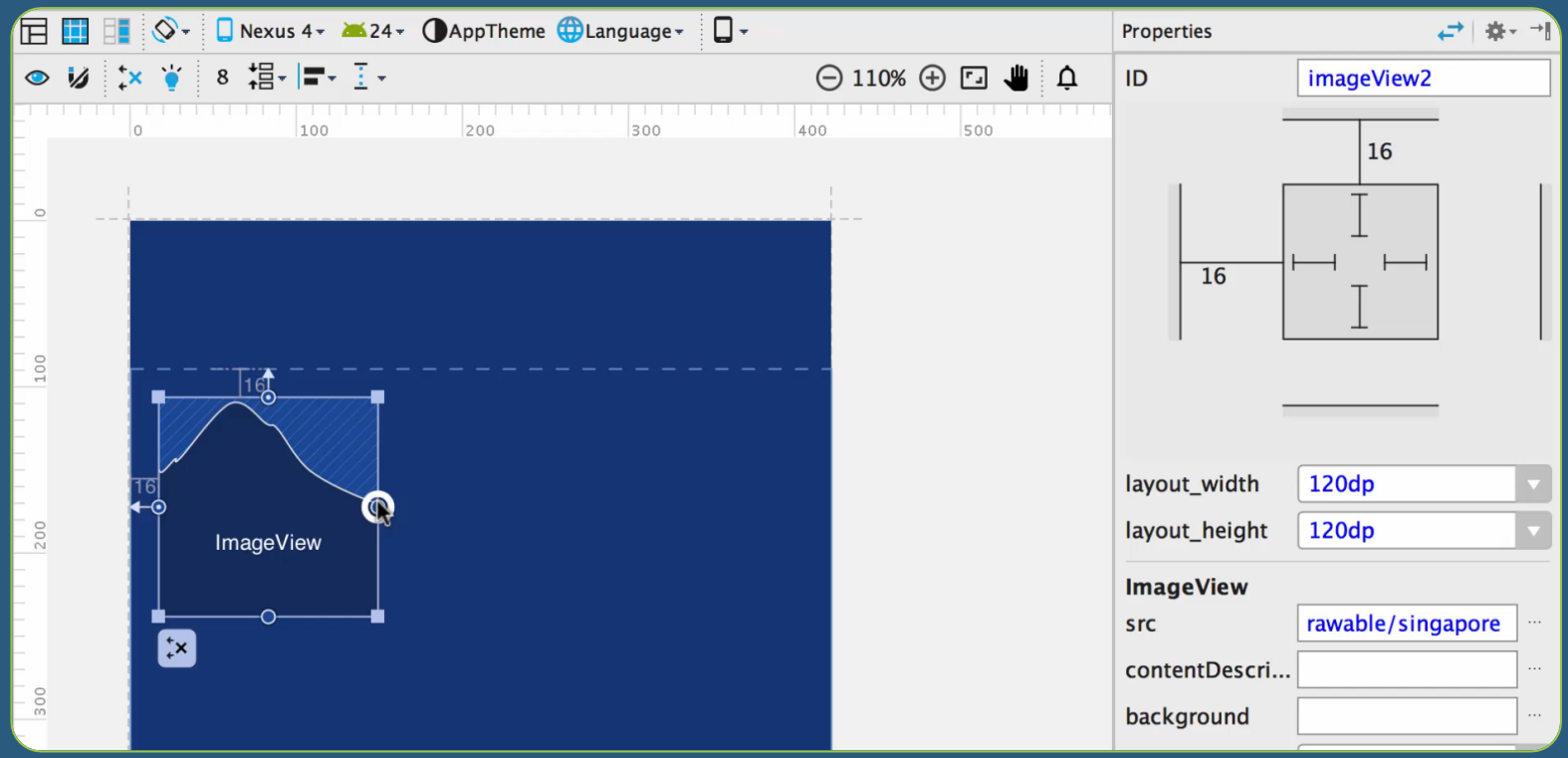


CONSTRAINT LAYOUT

- Adjust constraint bias
  - When you add a constraint to both sides of a view and the view size for the same direction is either “fixed” or “wrap content”, the view becomes centered between the two constraints with a default bias of 50%
  - The bias can be adjusted by dragging the bias slider in the “Properties” window or by dragging the view

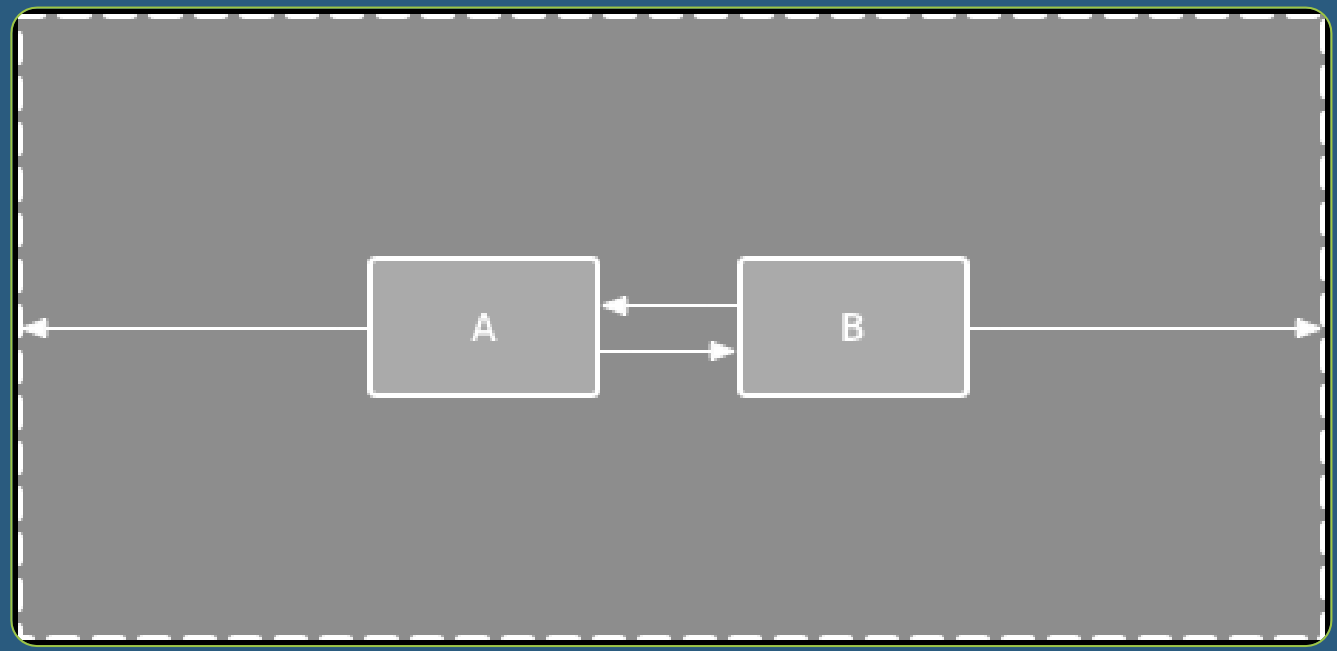
CONSTRAINT LAYOUT

- Adjust constraint bias



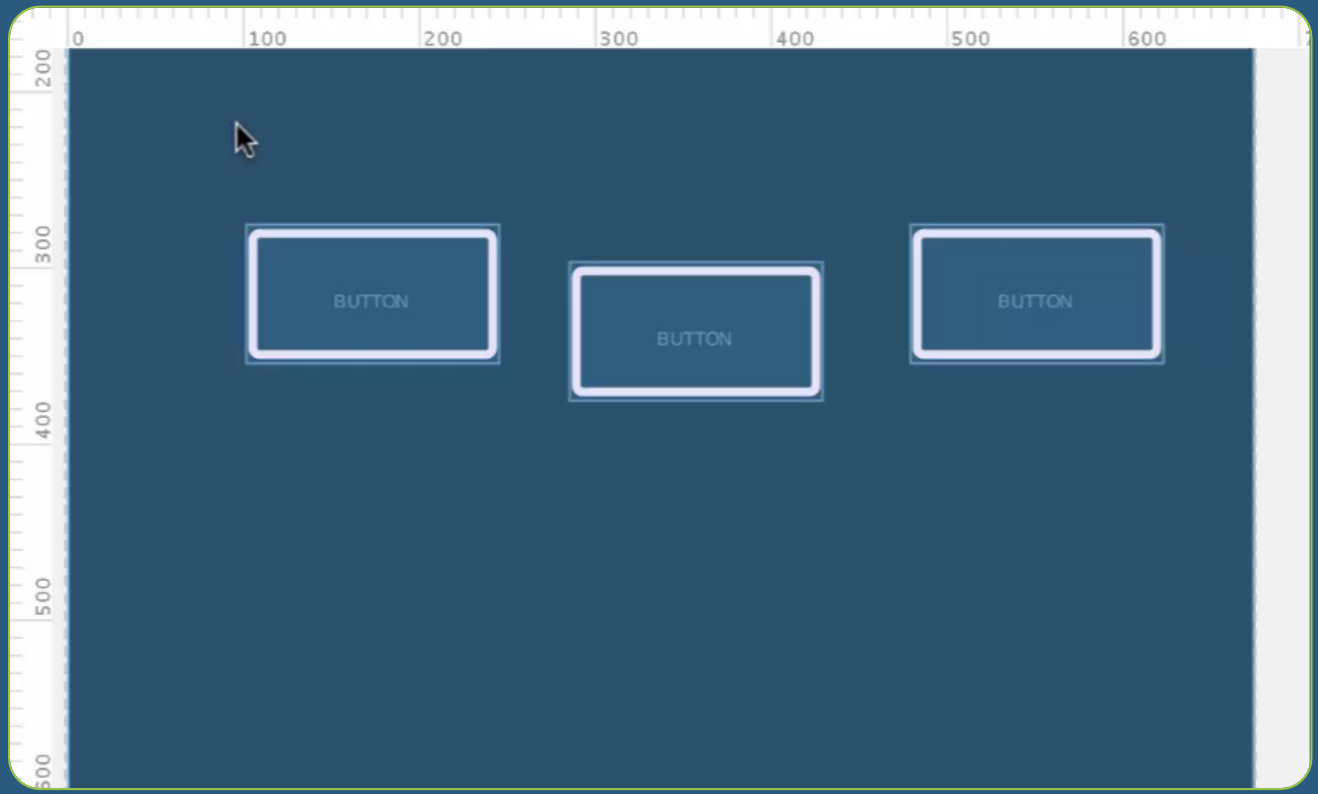
CONSTRAINT LAYOUT

- Control linear groups with a chain
- A chain is a group of views that are linked to each other with a bidirectional position constraint



CONSTRAINT LAYOUT

- To create a chain of views quickly, select them all, right-click one of the views, and then select either "Center Horizontally" or "Center Vertically" to create either a horizontal or vertical chain




loading