droidcon academy

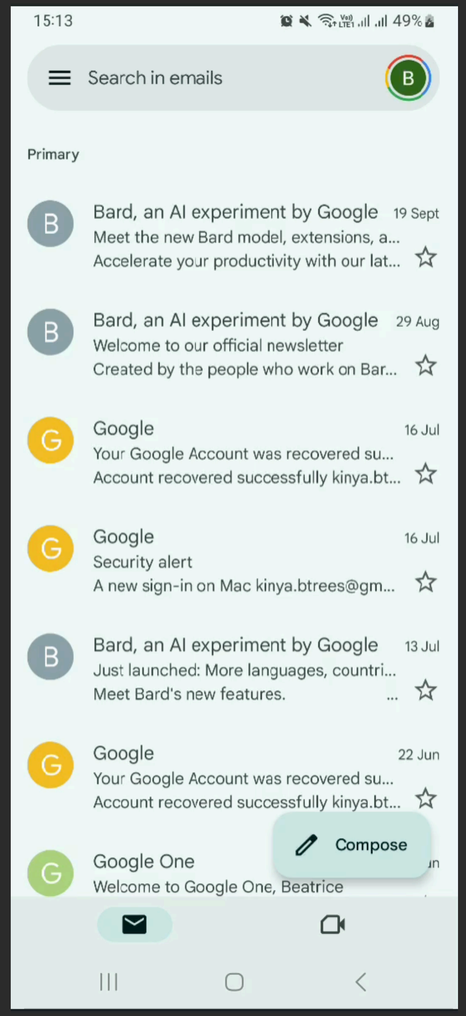
# Mastering Swipe to Action With Jetpack Compose in Android.


Beatrice Kinya

@B__Kinya

Beatrice Kinya

# Introduction



# Overview

- AnchoredDraggable modifier.
- AnchoredDraggableState
- Anchors/stop points that depend on the screen size available
- Saving state through configuration changes
- Implementing swipe-to-action gesture using Jetpack compose

droidcon academy

# AnchoredDraggable Modifier

- AnchoredDraggable modifier allows you to drag content either vertically or horizontally.
- The modifier was released with Jetpack Compose Foundation 1.6.0-alpha01.
- It supports advanced use cases such as anchors that depend on size of a component.
- AnchoredDraggable modifier has to main parts:
  - The Modifier that is applied to the content to be dragged
  - AnchoredDraggableState: State that specifies how drag will operate.

# AnchoredDraggableState

```
class AnchoredDraggableState<T>(
    initialValue: T,
    internal val positionalThreshold: (totalDistance: Float) -> Float,
    internal val velocityThreshold: () -> Float,
    val animationSpec: AnimationSpec<Float>,
    internal val confirmValueChange: (newValue: T) -> Boolean = { true }
)
```

droidcon academy

# AnchoredDraggableState parameters

- initialValue : Used to set the starting point of a draggable content.
- positionalThreshold : Determines whether an element will move to the next anchor or not based on the distance between the anchors.
- velocityThreshold : Specifies speed at which an element should be moving when you release it for it to snap at new position.
- animationSpec : Used to determine how to animate draggable content.
- confirmValueChange : This is an optional callback that use to control whether state change should happen or not.

droidcon academy

# Additional APIs

- updateAnchors().
- requiredOffset().

droidcon academy

# updateAnchors

- updateAnchors() is used to specify stop points in the drag area.
- You are required to specify at least 2 anchors but you can as many as needed.
- DraggableAnchors is a helper method which for creating anchors.

```
fun updateAnchors(
    newAnchors: DraggableAnchors<T>,
    newTarget: T = if (!offset.isNaN()) {
        newAnchors.closestAnchor(offset) ?:
targetValue
    } else targetValue
)

fun <T : Any> DraggableAnchors(
    builder: DraggableAnchorsConfig<T>.()
-> Unit
): DraggableAnchors<T> = …
```

# AnchoredDraggable Modifier

```
fun <T> Modifier.anchoredDraggable(
    state: AnchoredDraggableState<T>,
    orientation: Orientation,
    enabled: Boolean = true,
    reverseDirection: Boolean = false,
    interactionSource: MutableInteractionSource? = null
)
```

droidcon academy

# AnchoredDraggable Modifier Parameters

- state: This is an instance of AnchoredDraggableState.
- orientation: to specify the direction to drag the content either vertically or horizontally.

# Up Next

# requiredOffset

- AnchoredDraggable modifier does not move content. It only calculates a new offset when user drags elements on the screen.
- requiredOffset(): Float returns the current offset of the draggable content.
- Apply the new offset to the content using Modifier.offset to move the content when drag state changes.

# Summary

- AnchoredDraggable modifier and AnchoredDraggableState class.
- updateAnchors and requiredOffset APIs.
- Customisation options like positionalThreshold, velocityThreshold and animation
- Making content draggable with fixed anchors points
- Creating anchor points that depend on the screen width or height available.
- Implementing vertical and horizontal drag.
- Saving state across configuration changes.

droidcon academy

# Implementing Swipe-to-Action Gesture

## Up Next

droidcon academy

# Summary

- AnchoredDraggable and offset modifiers.
- Implementing swipe to action.
- Using confirmValueChange lambda to control whether state change should happen or not.