Navigation in Android

Back, Up, Recent

Richard Fulcher, Android Interaction Designer
Adam Powell, Android Framework Engineer

Tasks in Android

A quick refresher

What is a Task?

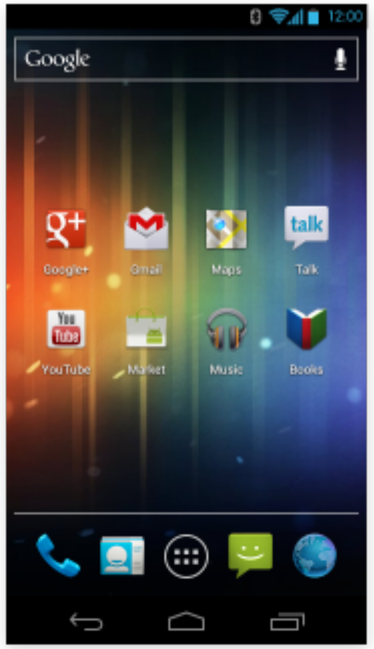
- A task is a **stack of activities**
- A task may include activities **from several apps**
    - (and one app's activities may appear in several tasks)
- A task is **not a process**

A simple Task

Open Gmail from Home, view conversation

Launcher

Task Stack

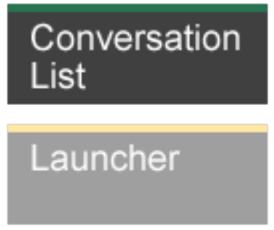


Home

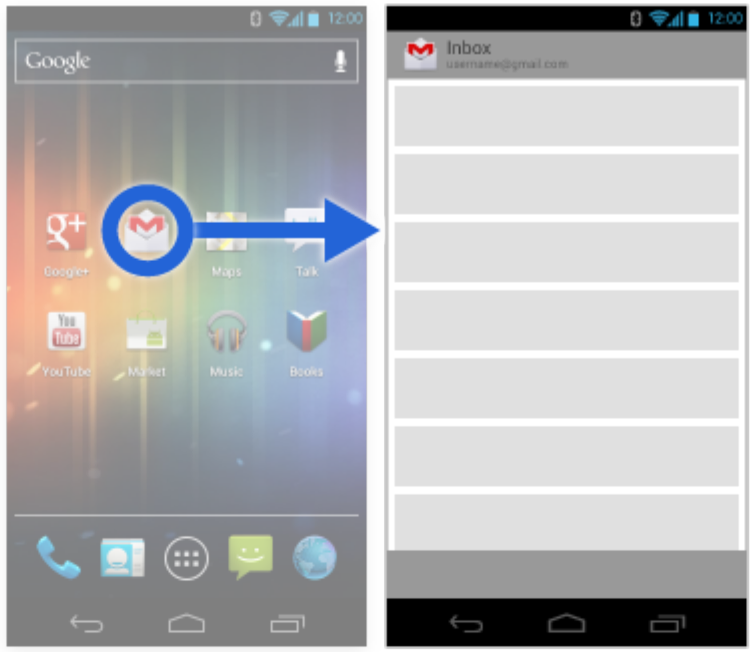
Launcher

A simple Task

Open Gmail from Home, view conversation



Task Stack

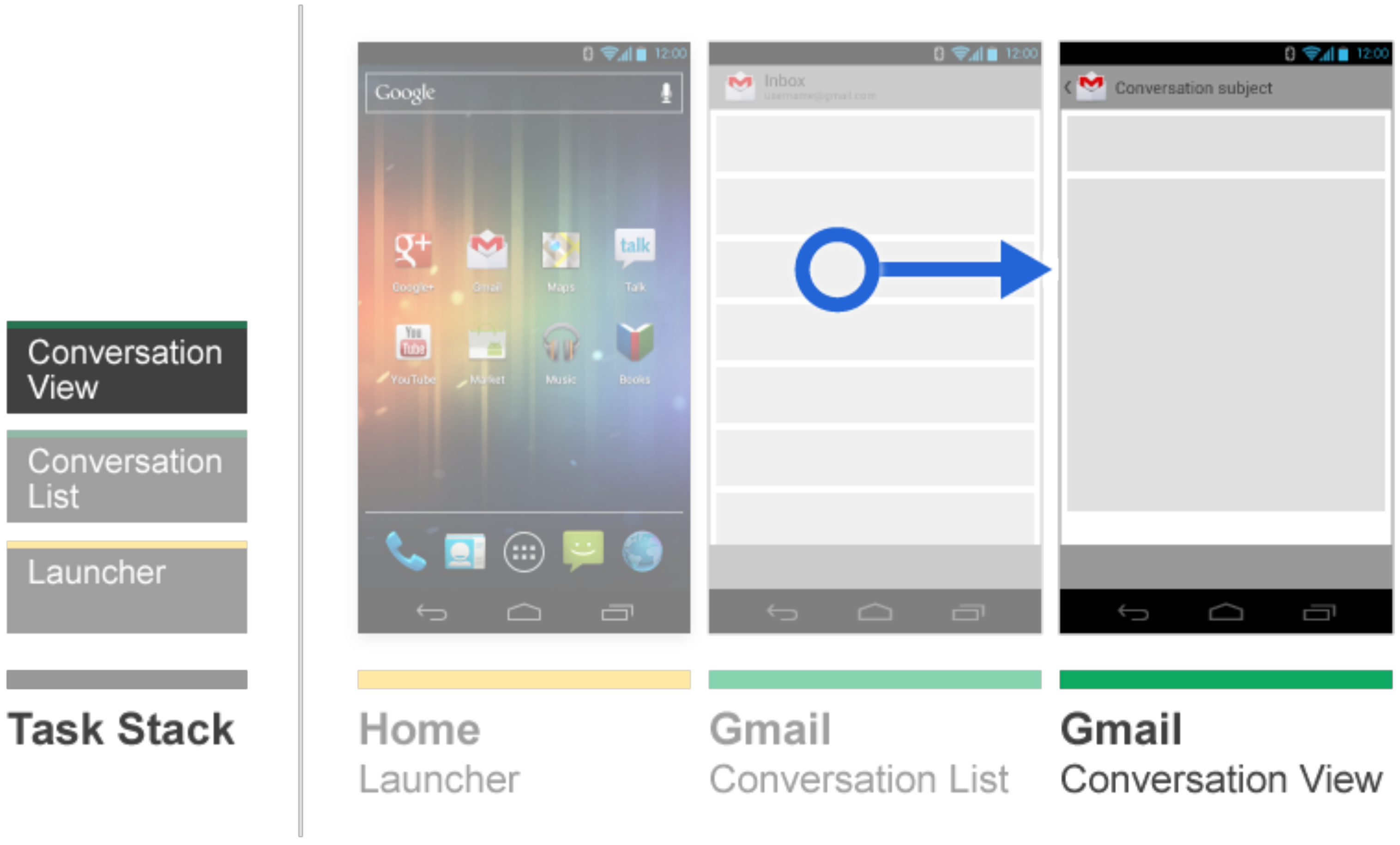


Home
Launcher

Gmail
Conversation List

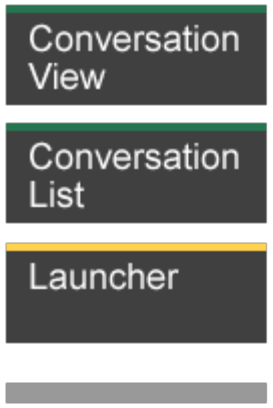
A simple Task

Open Gmail from Home, view conversation

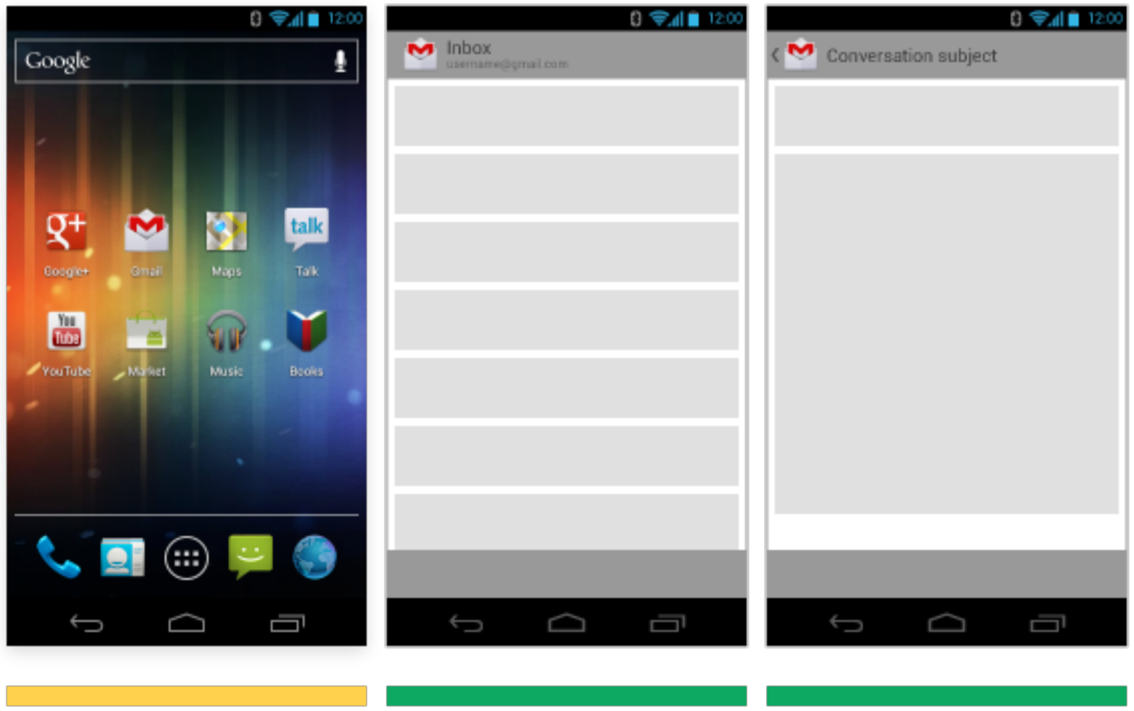


A simple Task

Open Gmail from Home, view conversation



Task Stack



Home
Launcher

Gmail
Conversation List

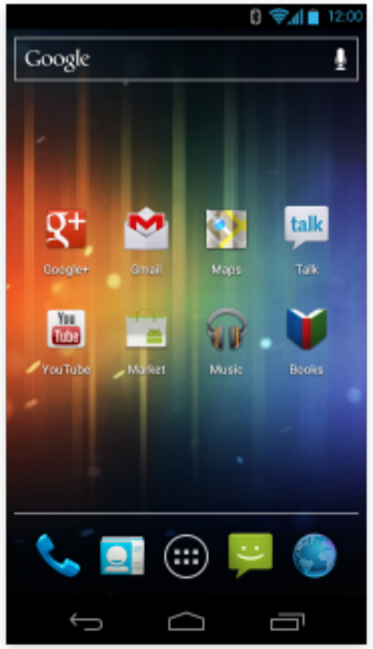
Gmail
Conversation View

A more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail

Launcher

Task Stack

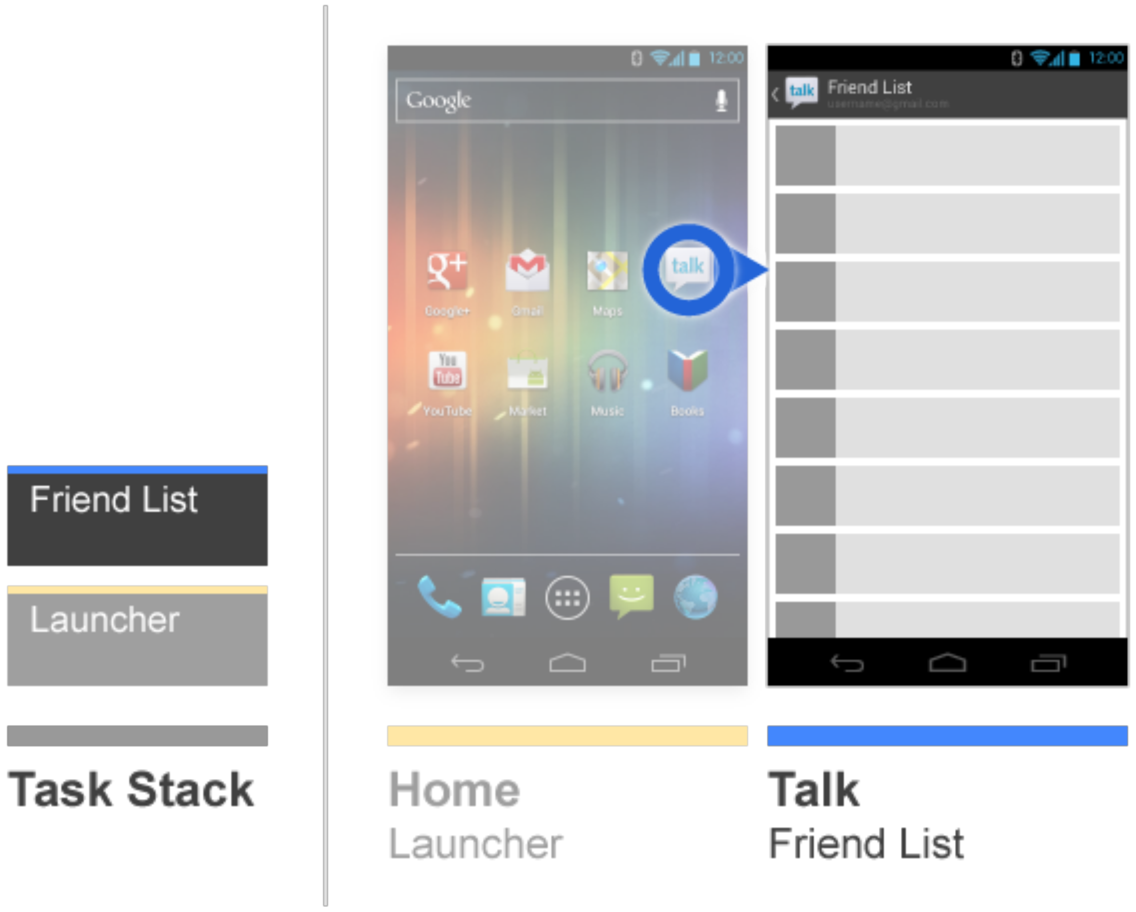


Home

Launcher

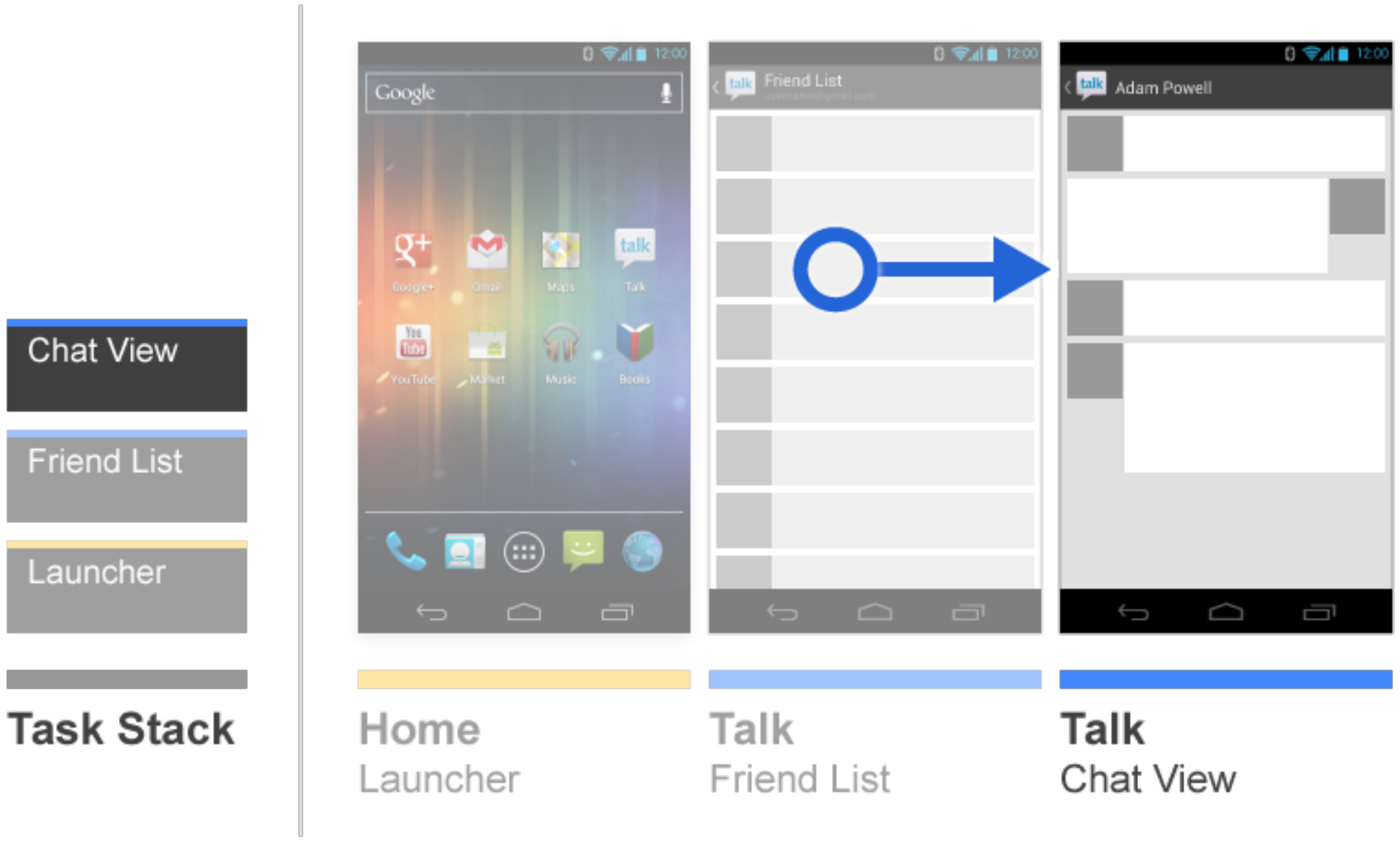
A more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail



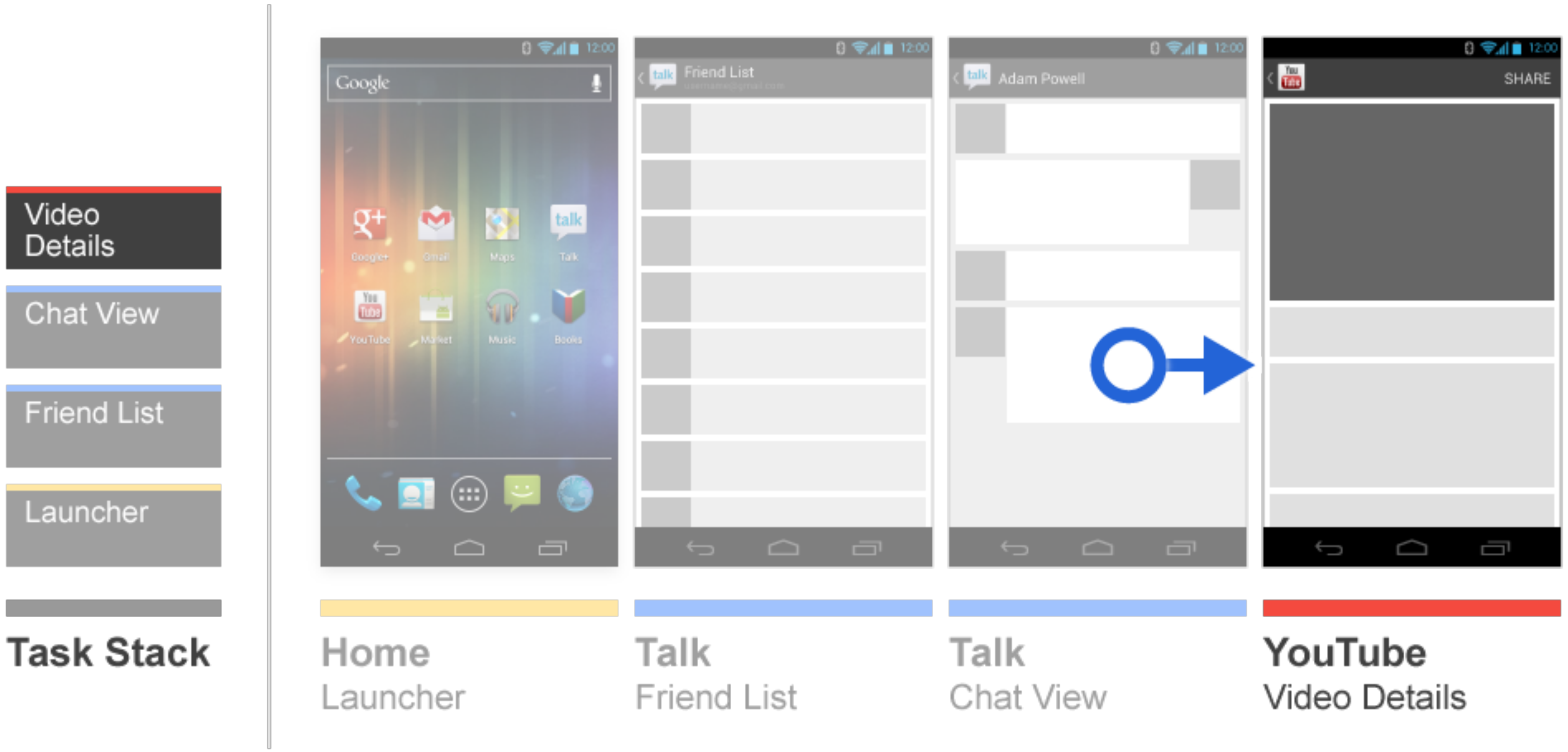
A more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail



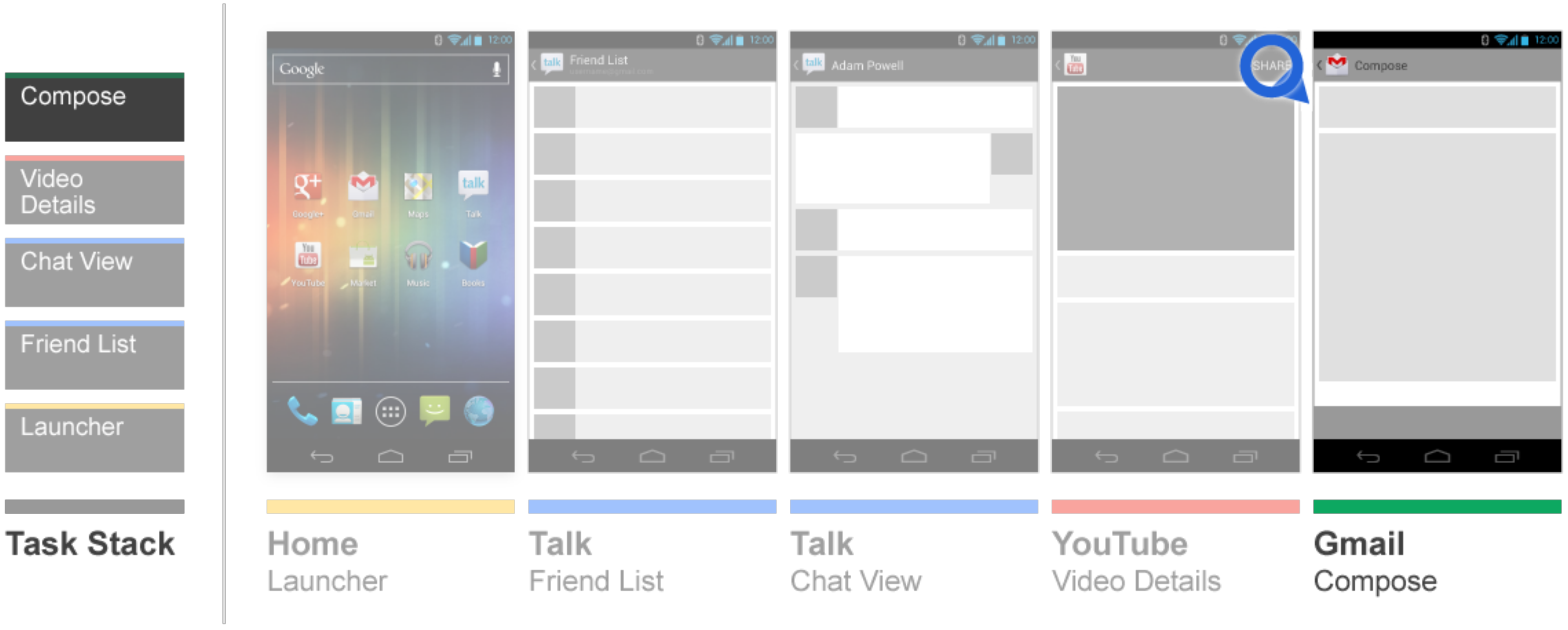
A more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail



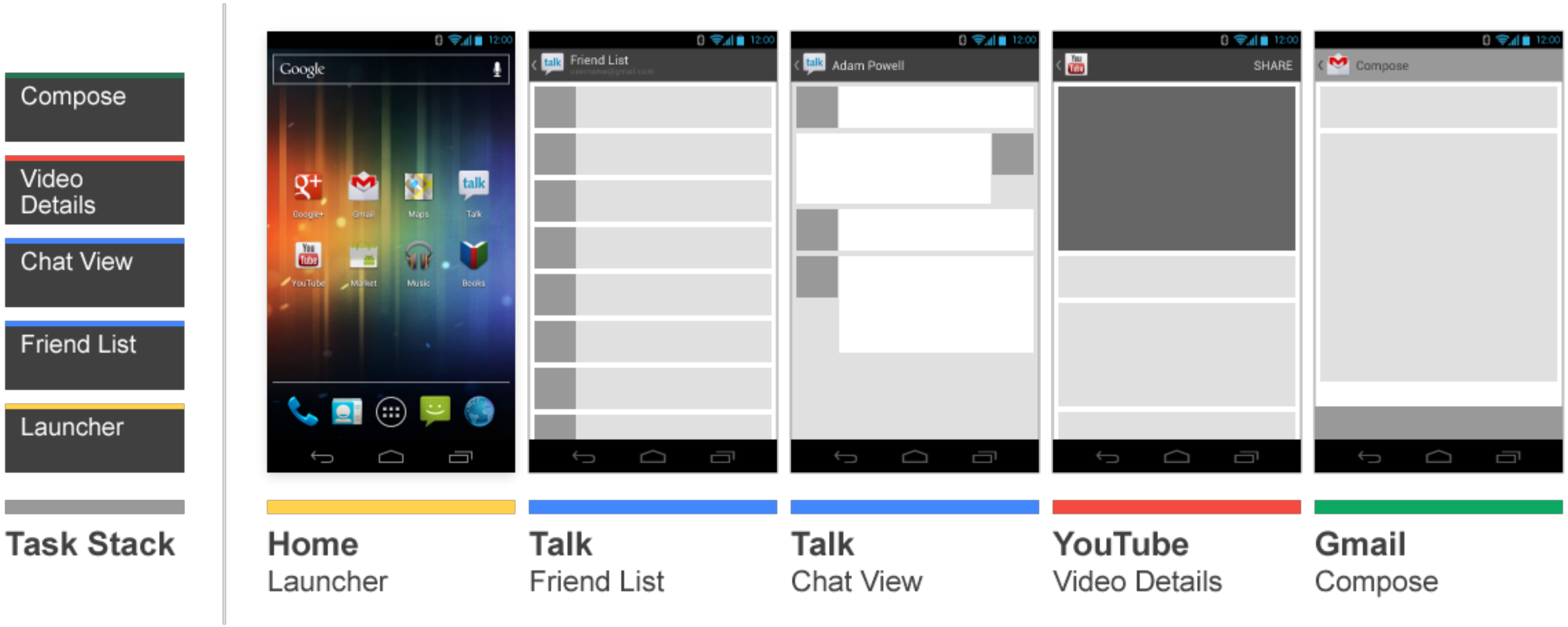
A more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail

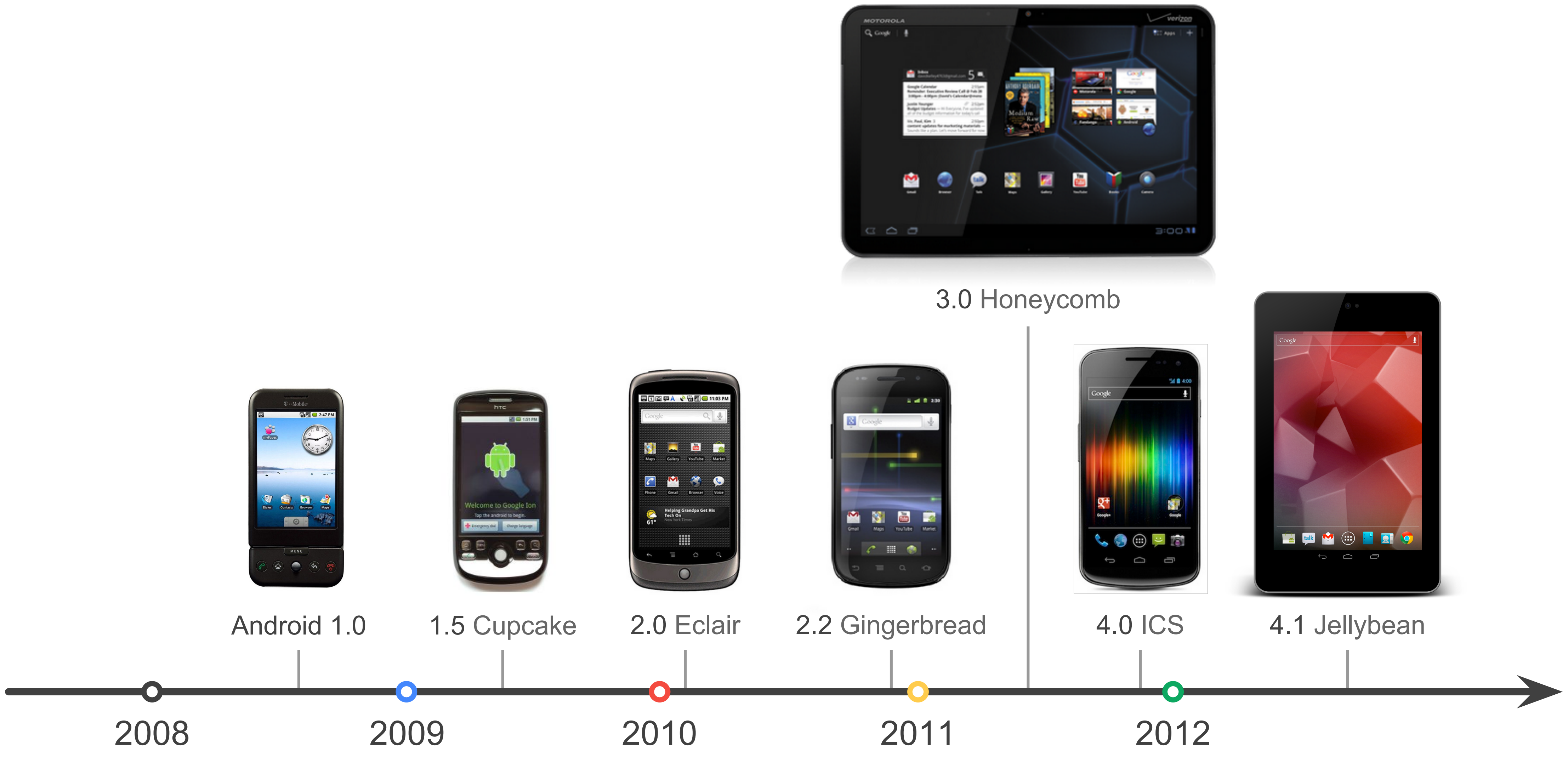


A more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail

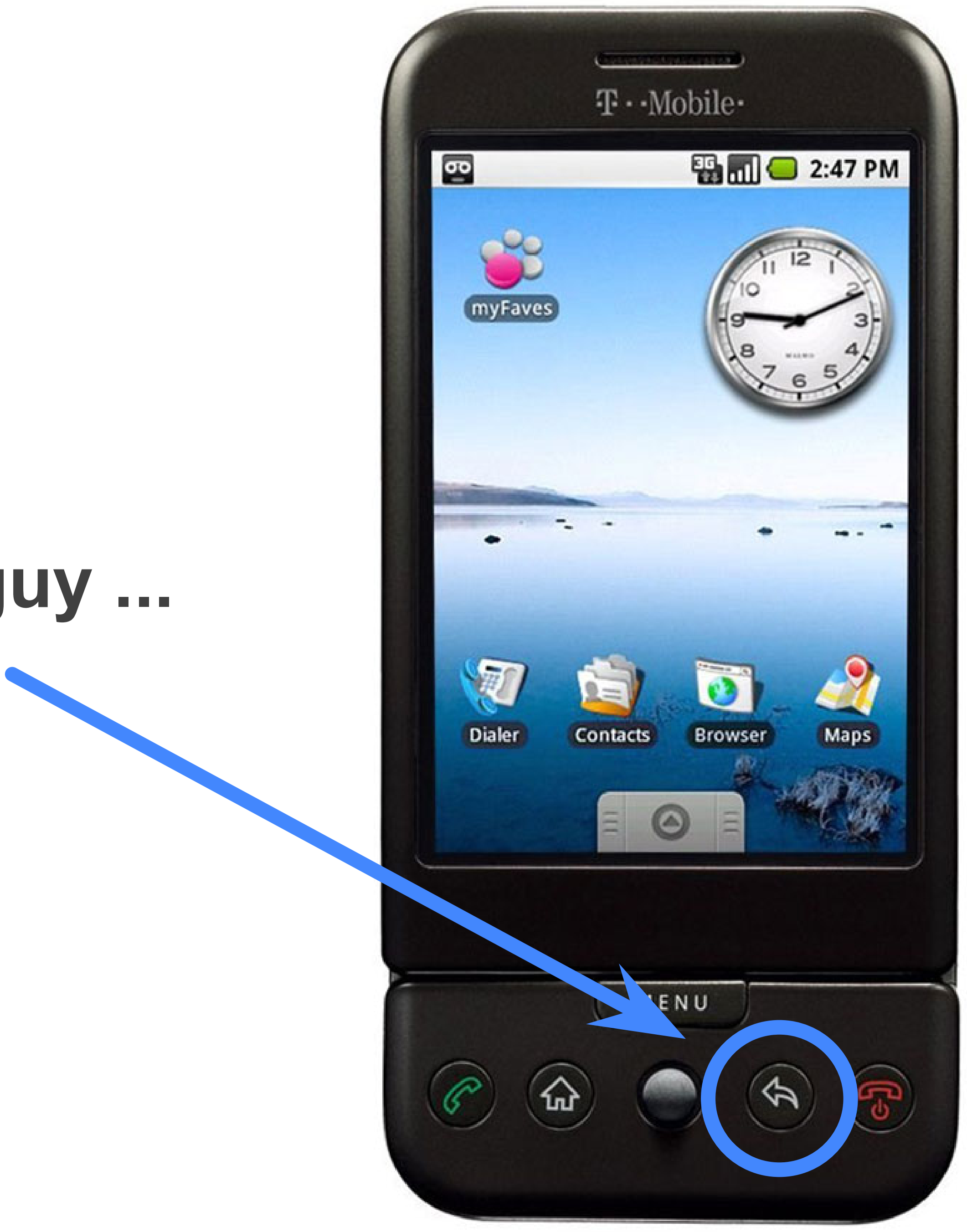


Tasks have always worked this way in Android

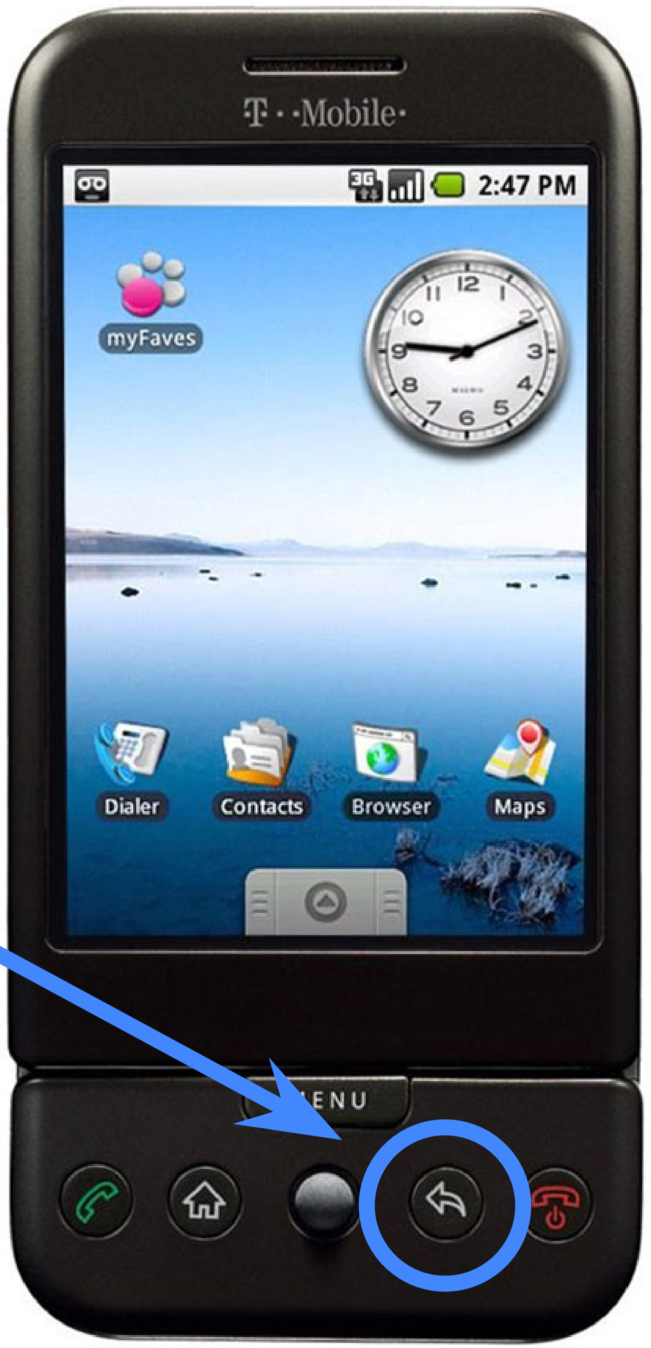


And this guy ...

And this guy ...



And this guy ...



... has been along for the whole ride

And this guy ...

... has been along for the whole ride

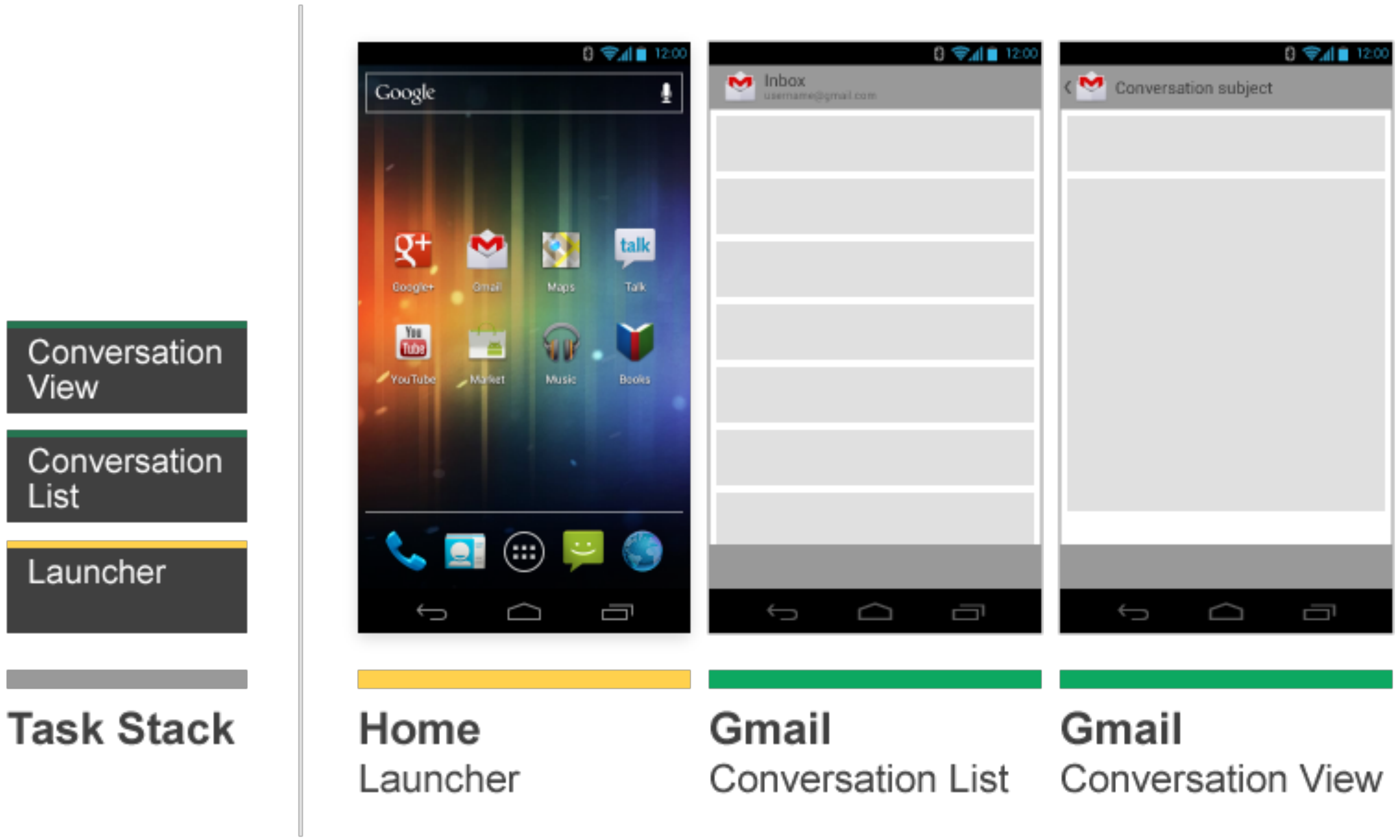
(though not without an occasional facelift)

How Back works

- Back unwinds the **global** activity history stack
- It **finish()es the current activity**
    - ActivityManager brings the previous activity in the stack forward
- With a few exceptions:
    - Closing the onscreen keyboard
    - Dismissing dialogs, spinners, etc.
    - Exiting contextual modes (e.g. selection)
    - Web browsing

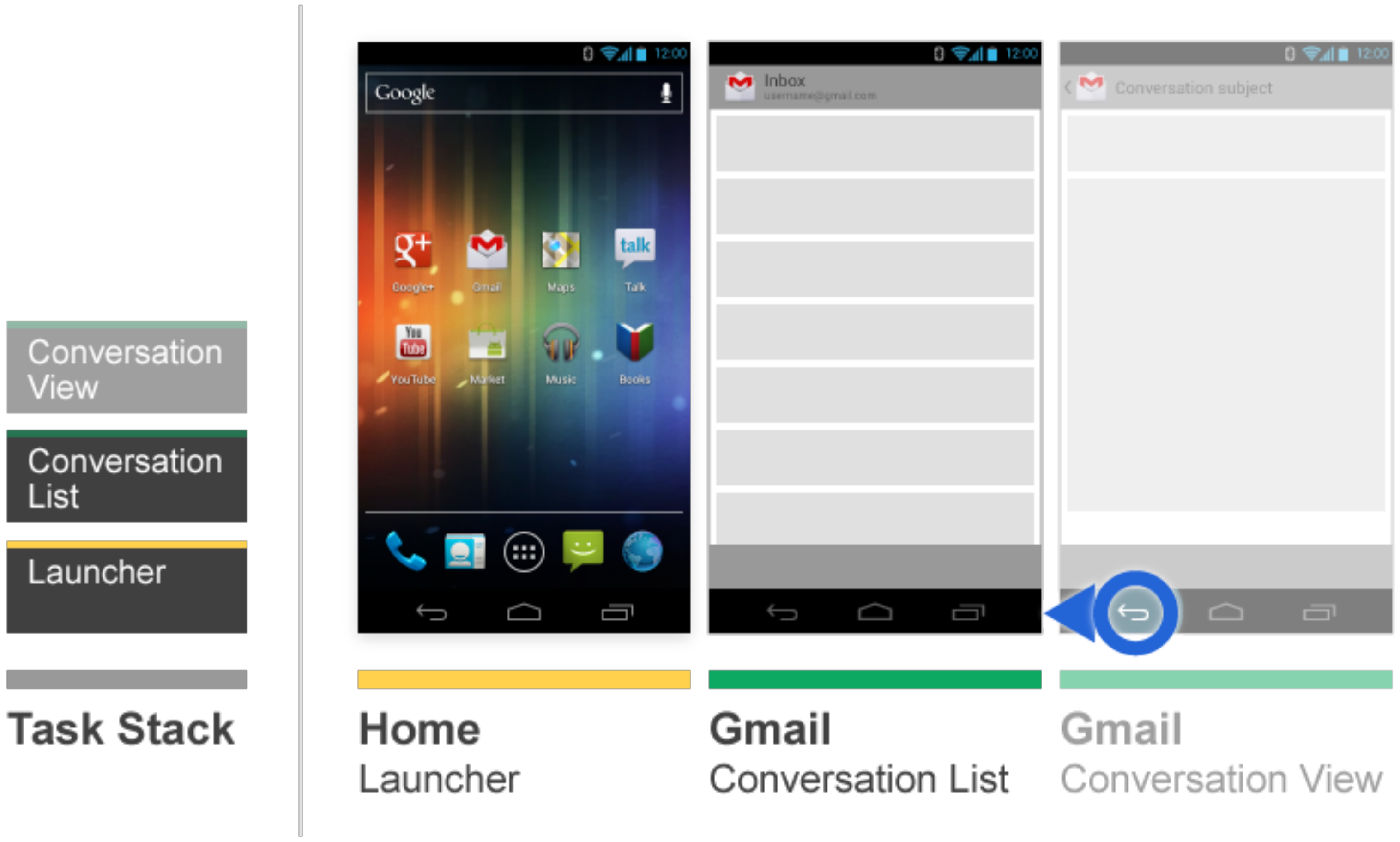
Back through a simple Task

Open Gmail from Home, view conversation



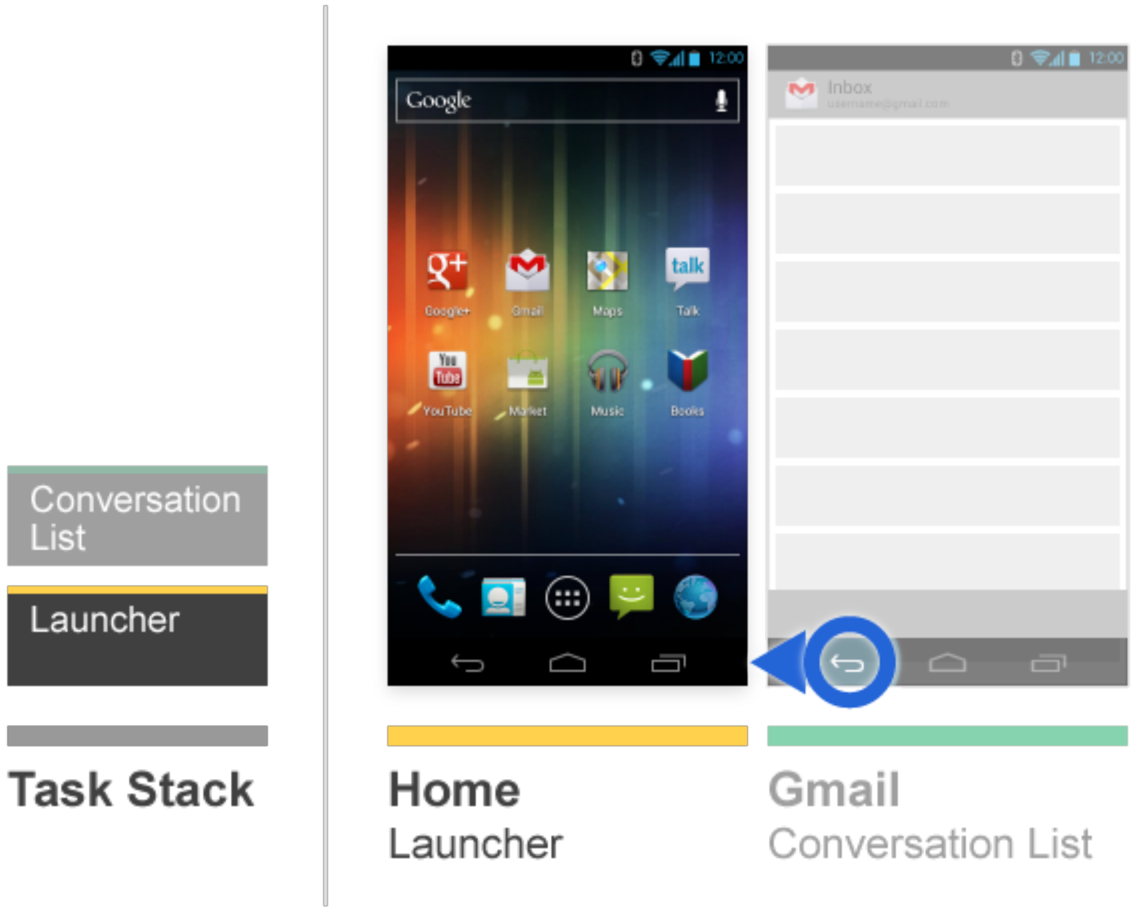
Back through a simple Task

Open Gmail from Home, view conversation



Back through a simple Task

Open Gmail from Home, view conversation

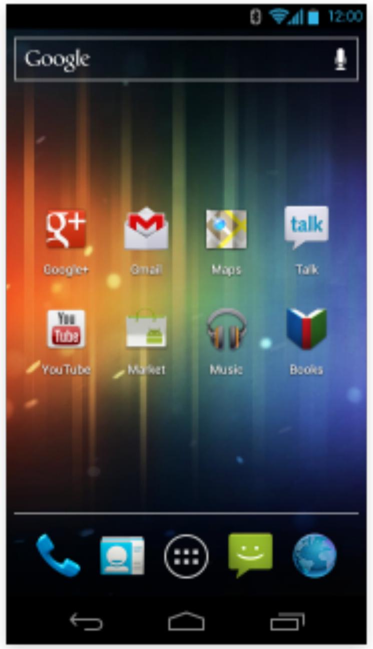


Back through a simple Task

Open Gmail from Home, view conversation

Launcher

Task Stack

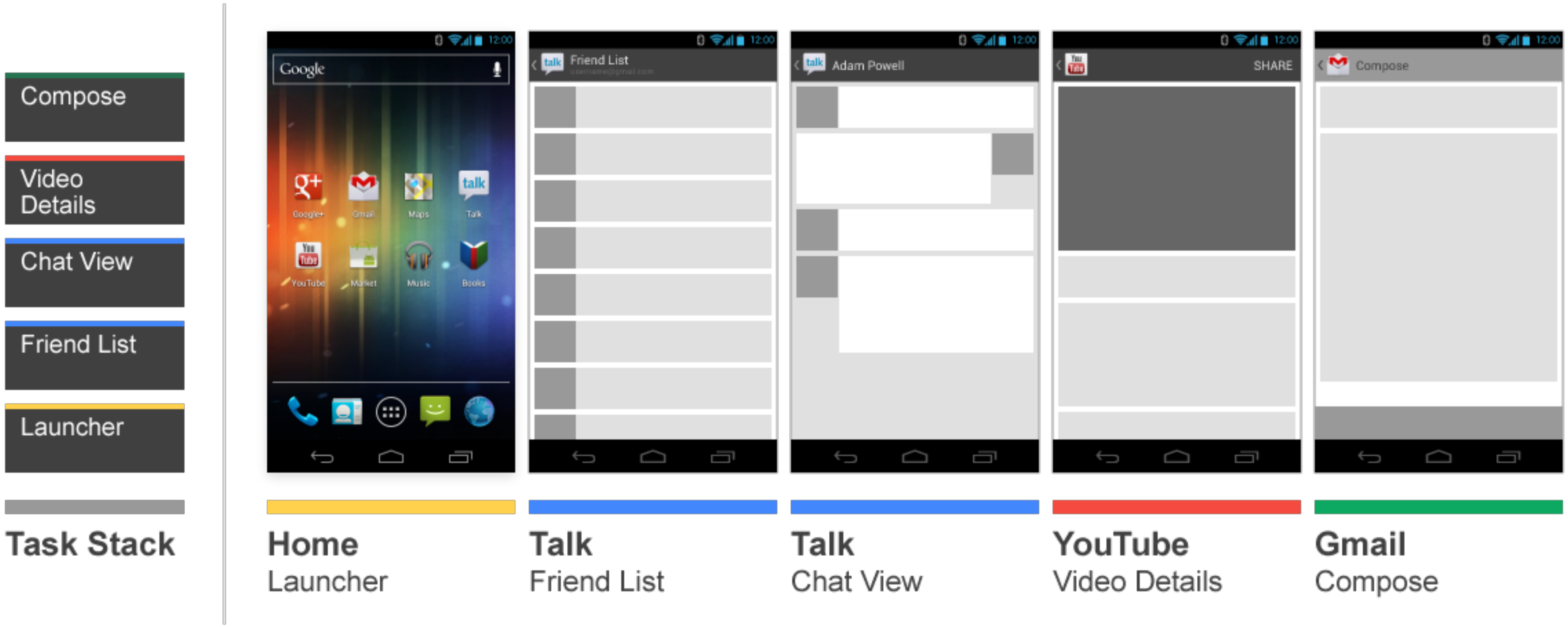


Home

Launcher

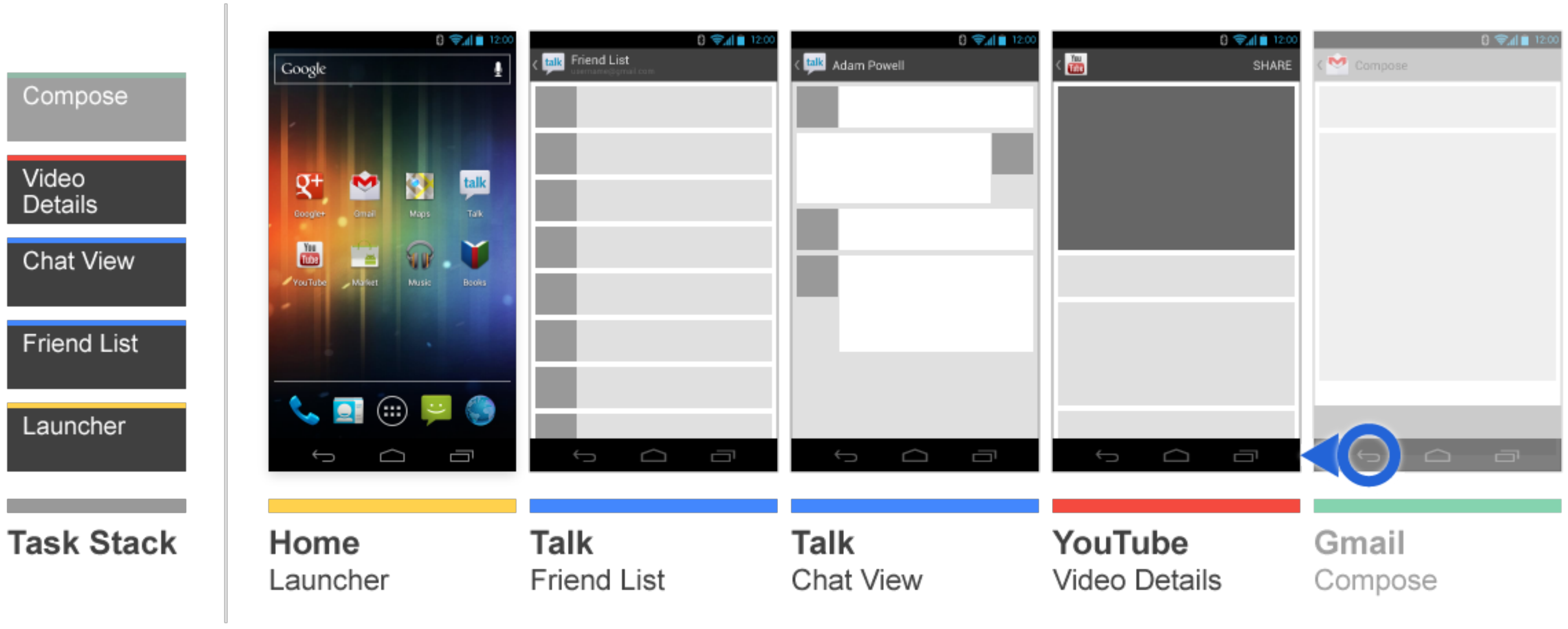
Back through a more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail



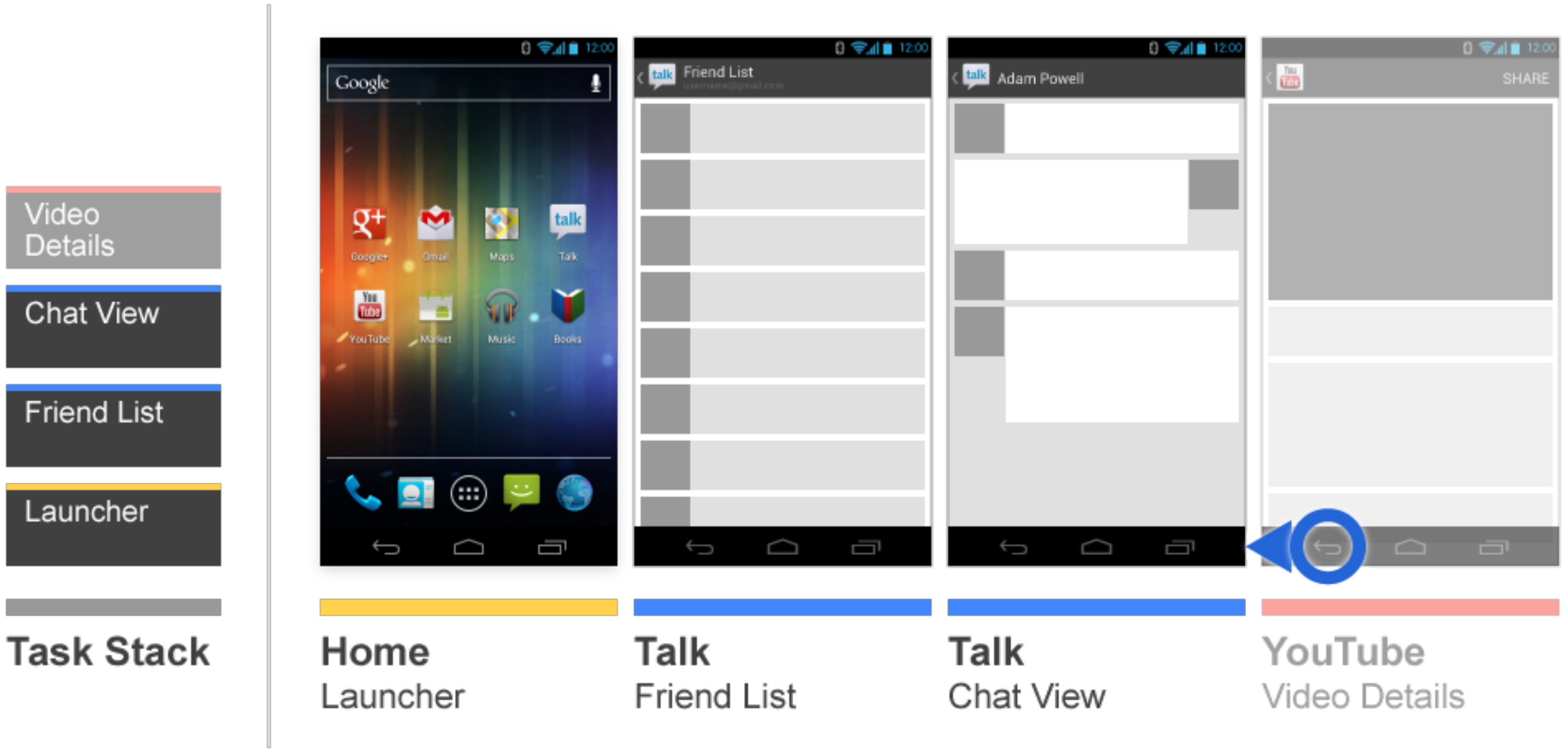
Back through a more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail



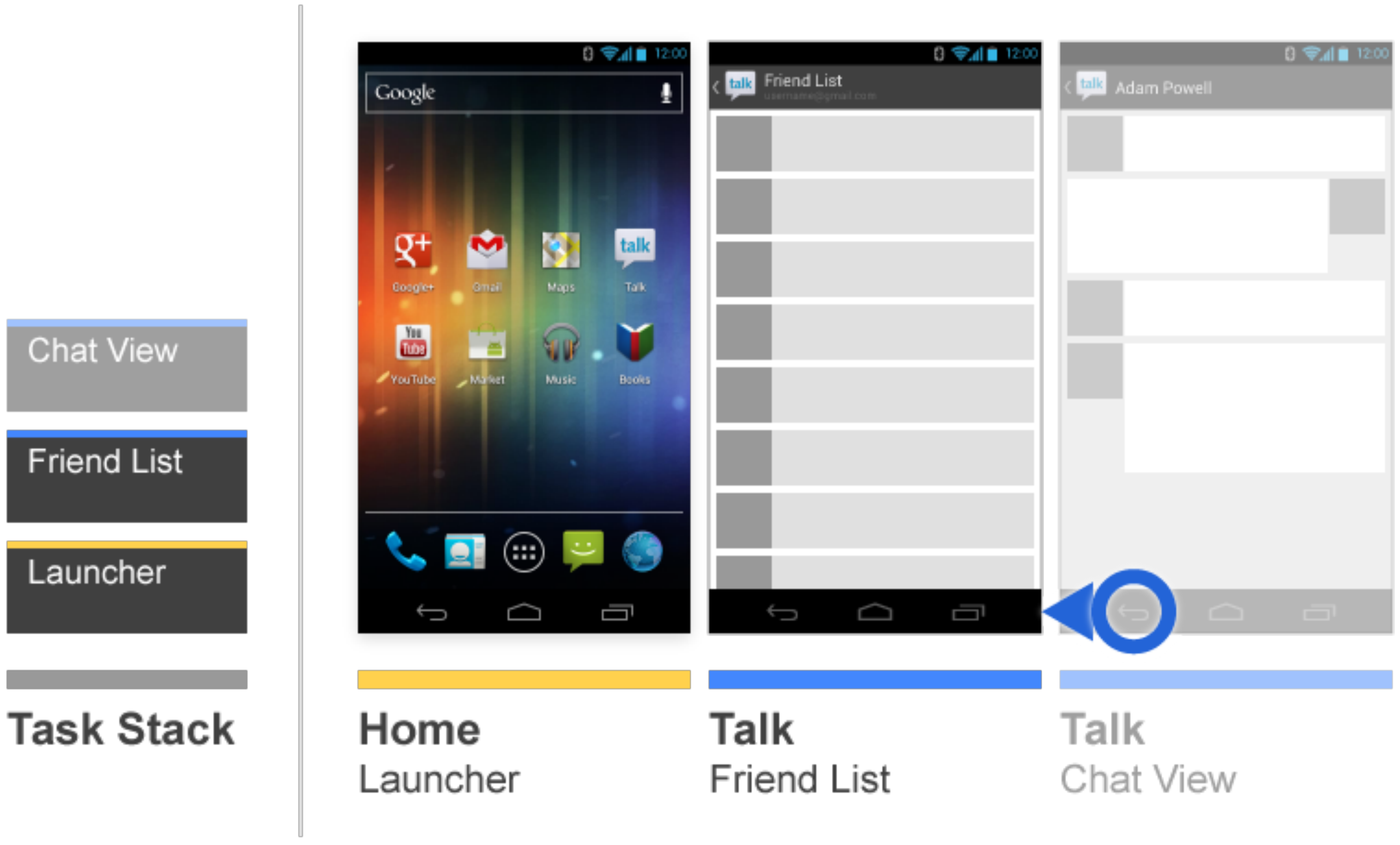
Back through a more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail



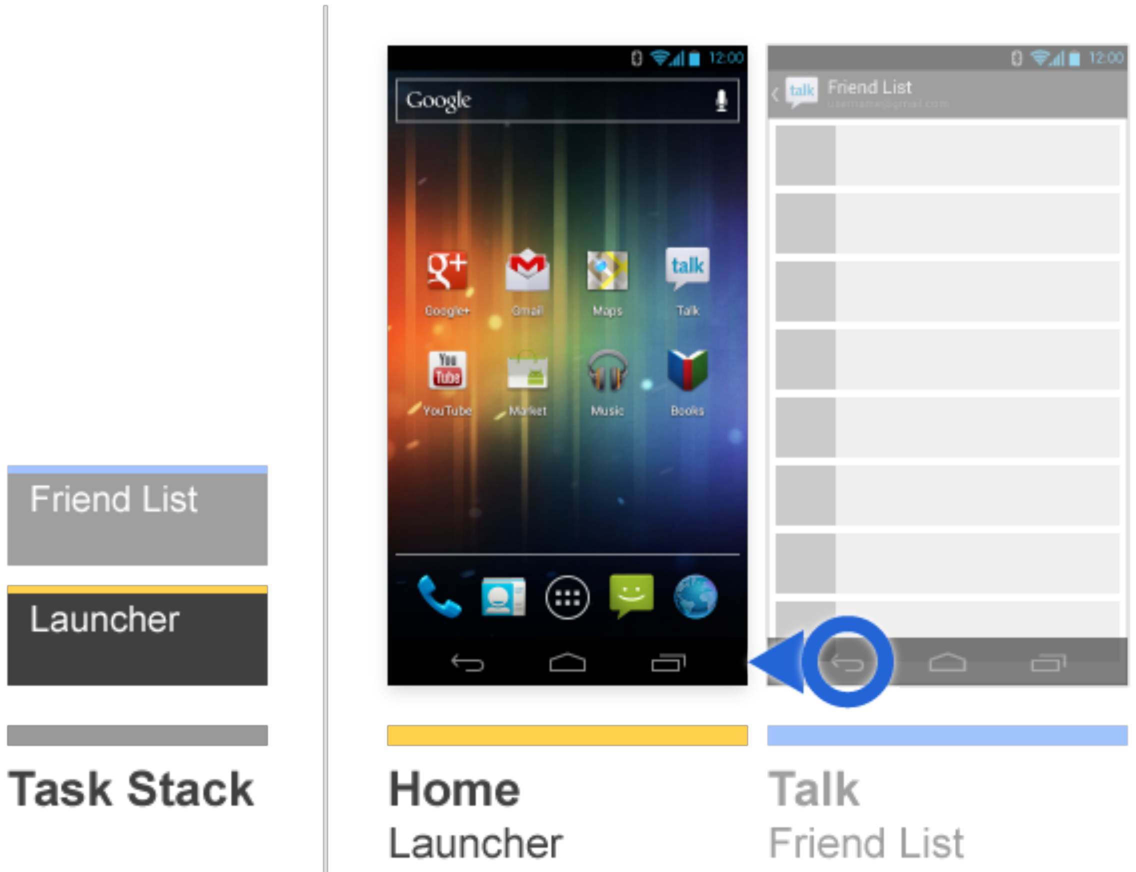
Back through a more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail



Back through a more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail

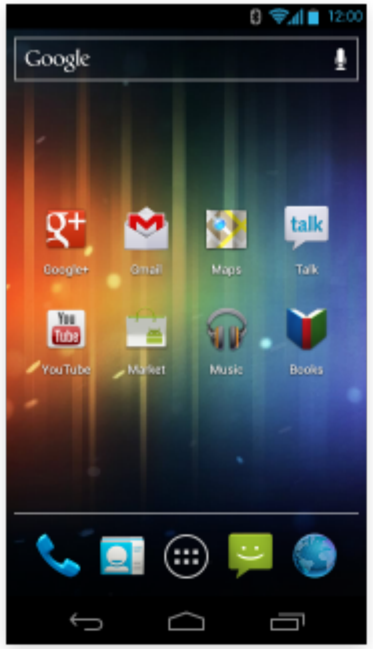


Back through a more complex Task

Open Talk, view Chat, view YouTube video, share video via Gmail

Launcher

Task Stack



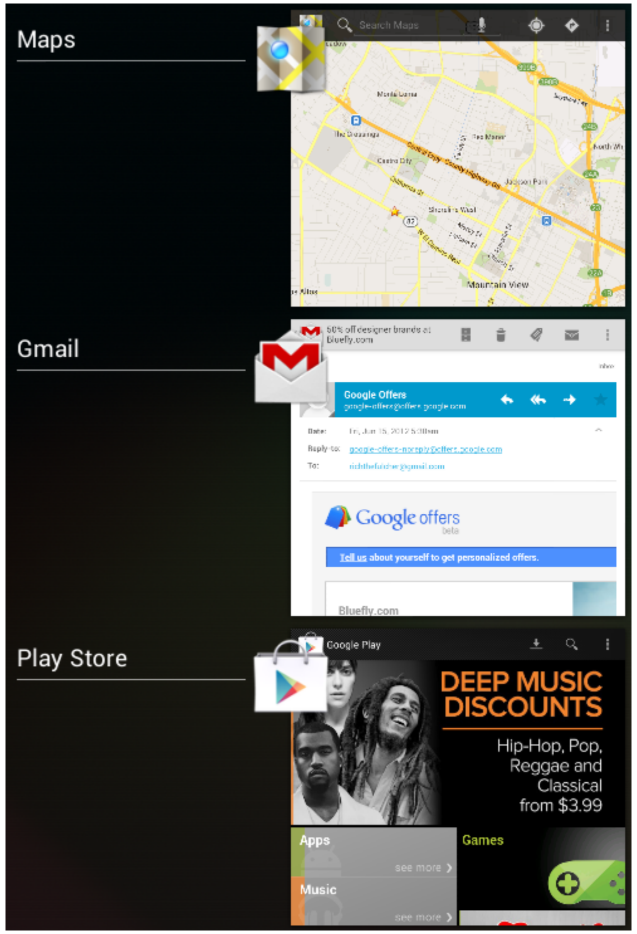
Home

Launcher

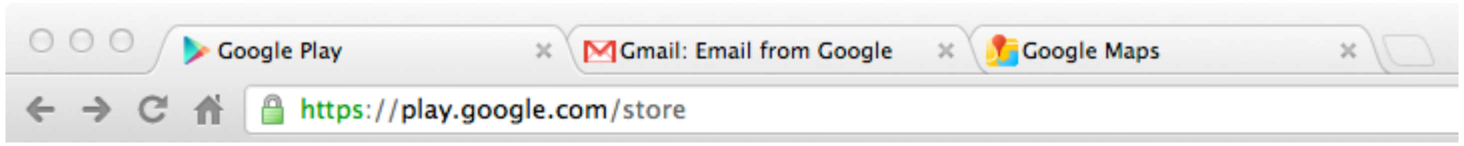
Multiple Tasks

Activity Manager and Task Affinity

Working with multiple Tasks



... is a lot like ...



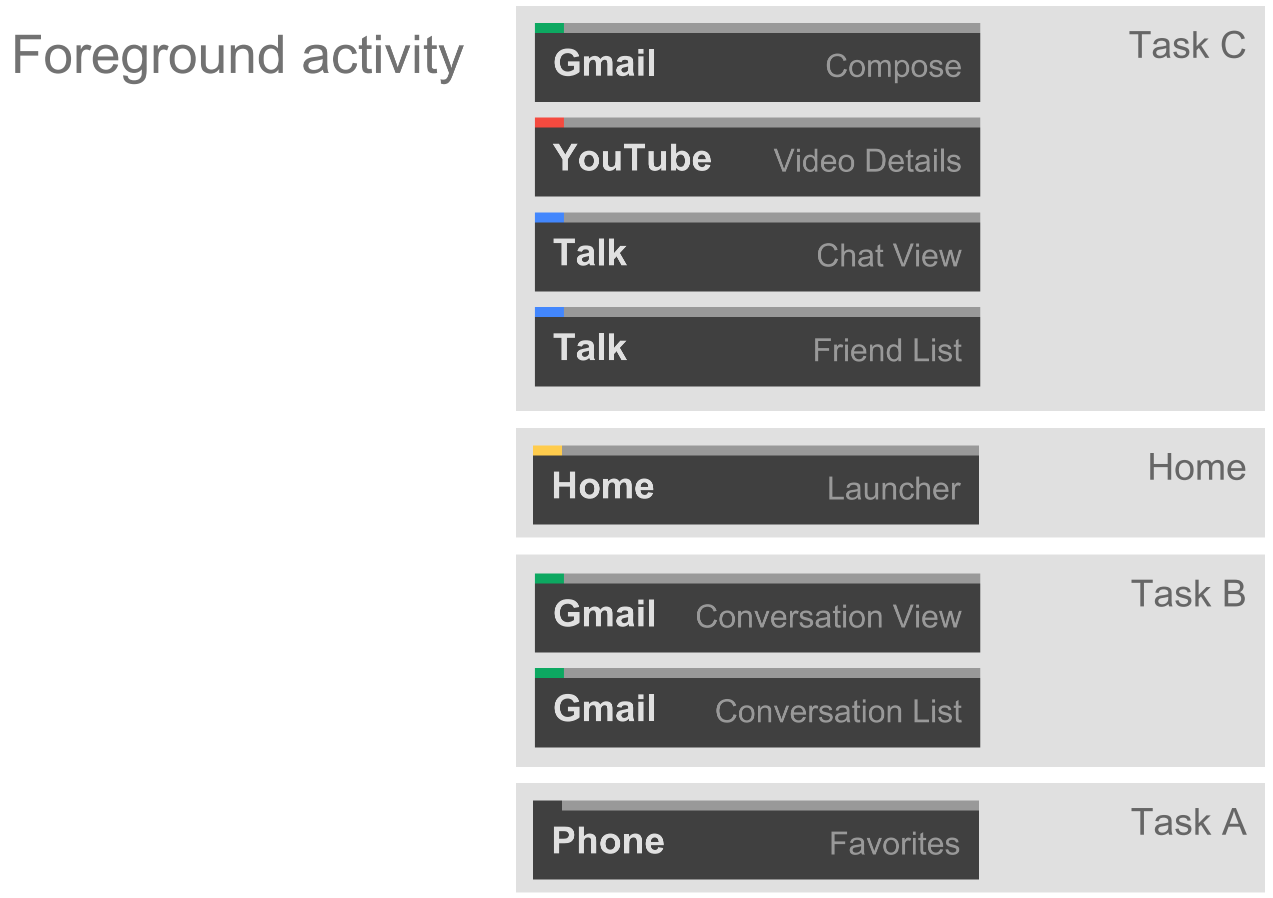
But how do tasks *really* work?

And what do all of those Intent flags mean?

```
Intent.FLAG_ACTIVITY_CLEAR_TASK

Intent.FLAG_ACTIVITY_NEW_TASK

Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET

Intent.FLAG_ACTIVITY_TASK_ON_HOME
```

How ActivityManager sees the world

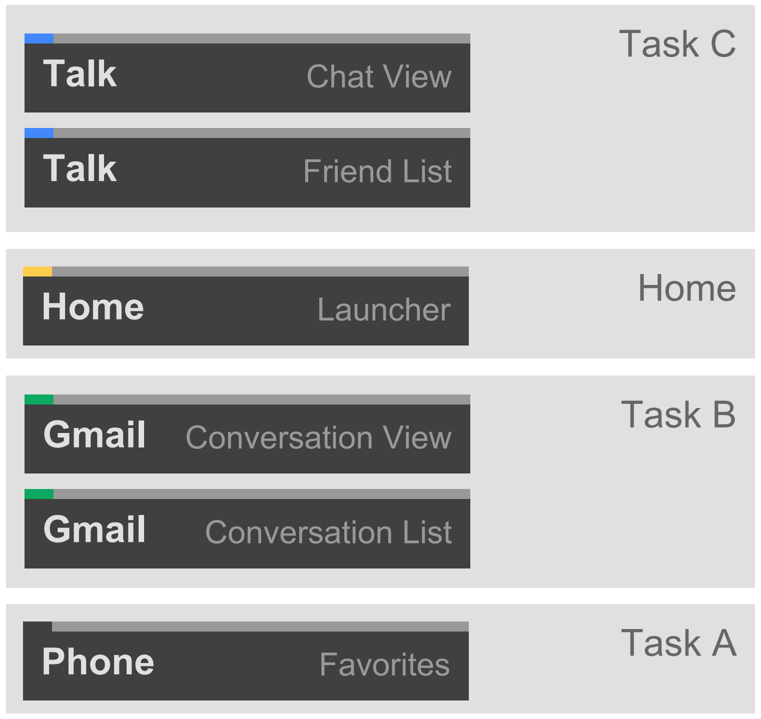
One big global activity stack!



Creating a new task

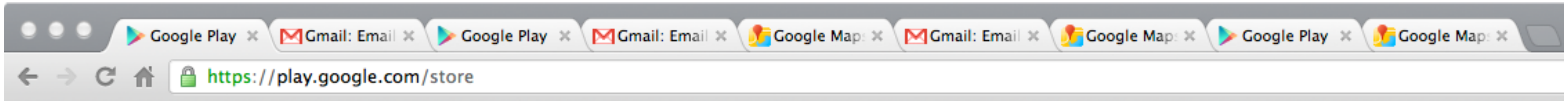
Jump tasks with Intent.FLAG_ACTIVITY_NEW_TASK

```
Intent target = new Intent(myActivity, TargetActivity.class);
target.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

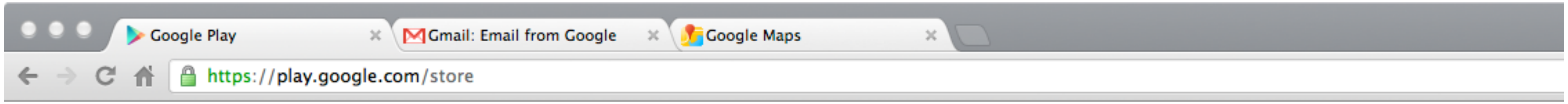


Avoiding the "open in new tab" problem

- Every Activity has a Task Affinity
  - Defined in your manifest on `<activity>` elements
  - Defaults to your package name
- Turns this:



- Into this:



When a task with the same affinity already exists

... and an Activity is started with `Intent.FLAG_ACTIVITY_NEW_TASK` ...

- The existing task is brought to the front

- If the Intent matches the root Intent of the existing task...
  - We're done!
  - The net effect was a simple task switch

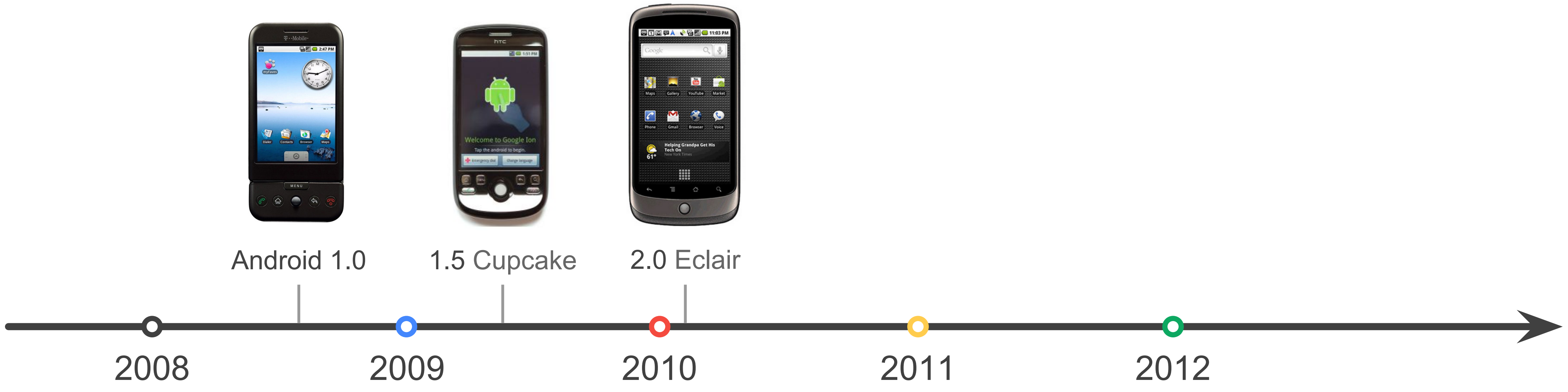
- If not...
  - The new Activity is started on top of the existing task

This is why Launcher can act like a task switcher

Navigation Evolves

The Emergence of Up and Recents

By the start of 2010, we knew we had a problem

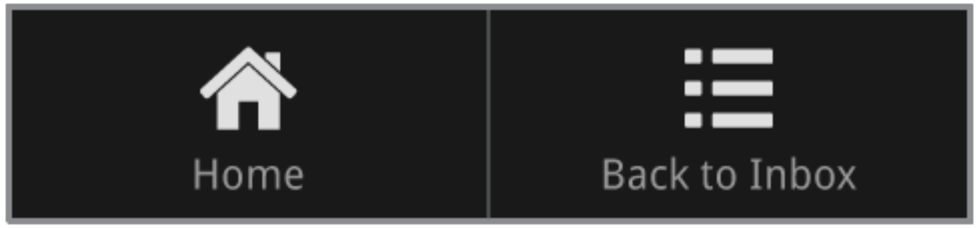


System-level navigation controls

Back

Home
+ Recents

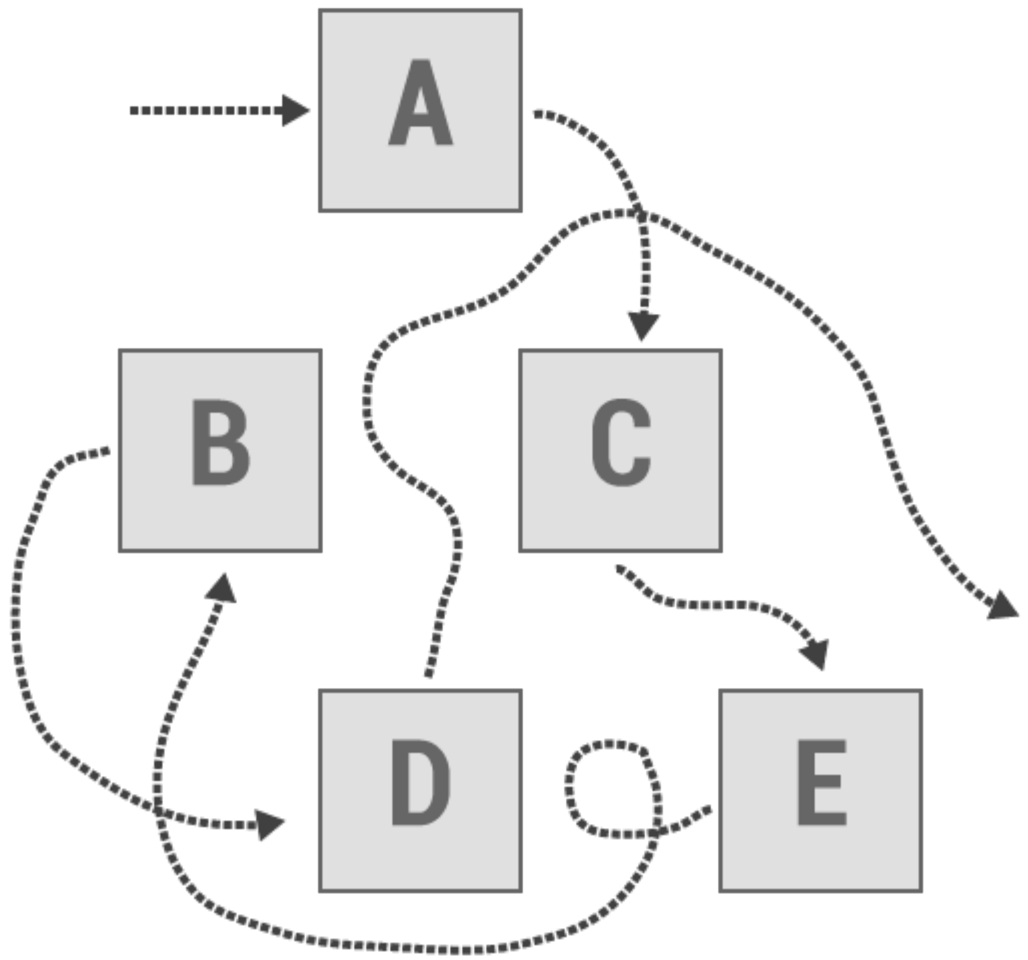
App-level navigation controls



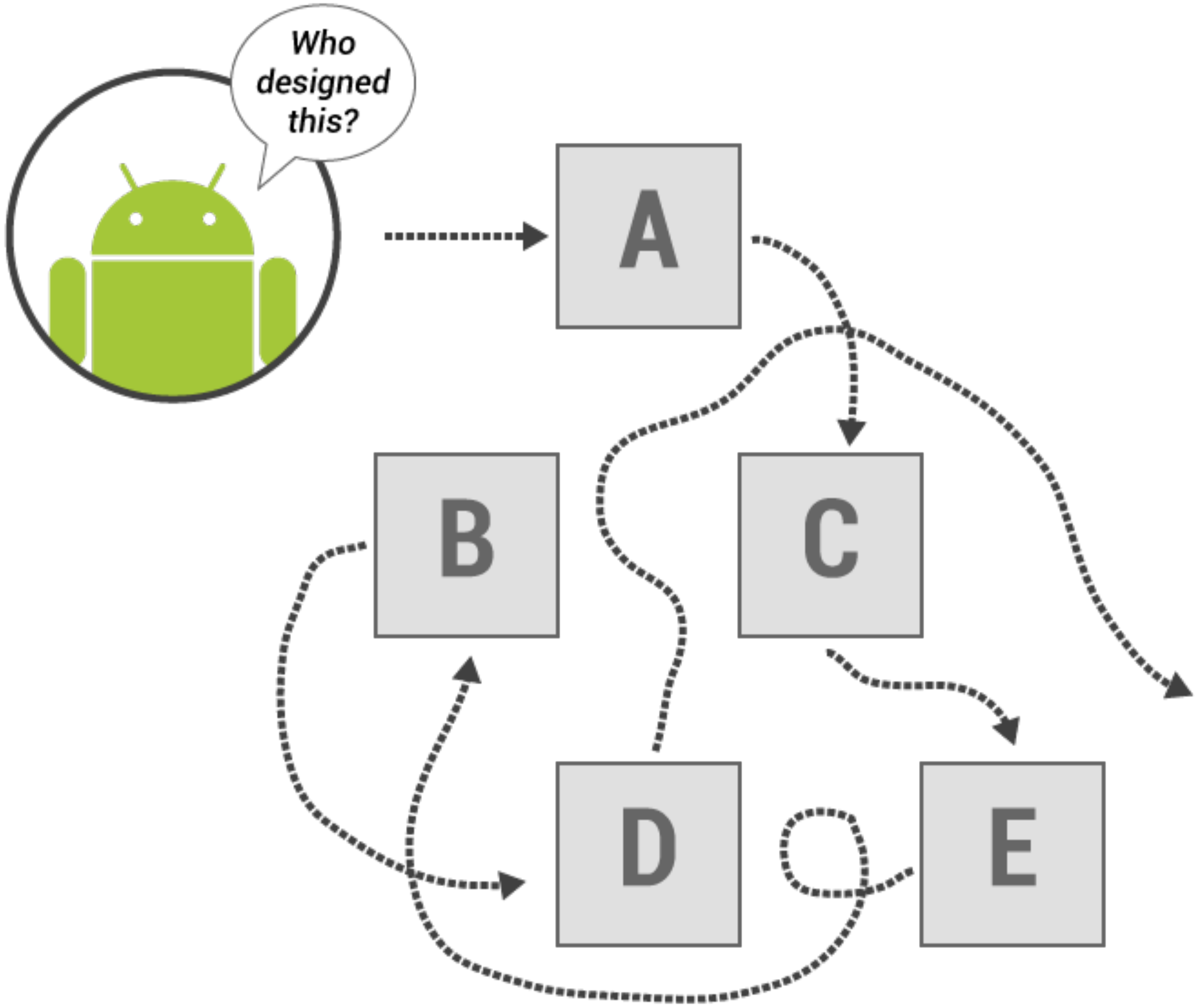
!

Menu (umm ...) Back

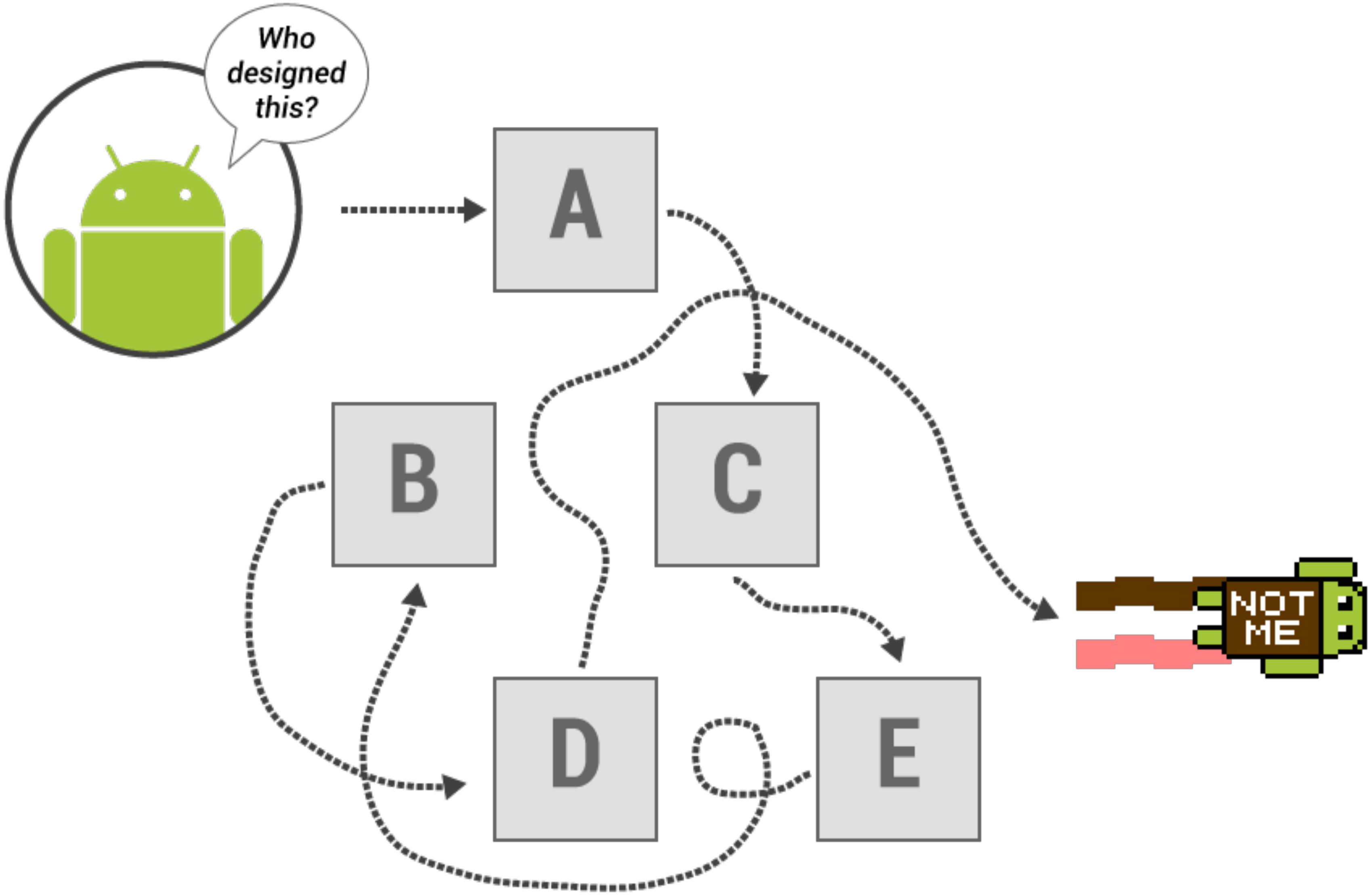
How did I wind up here?



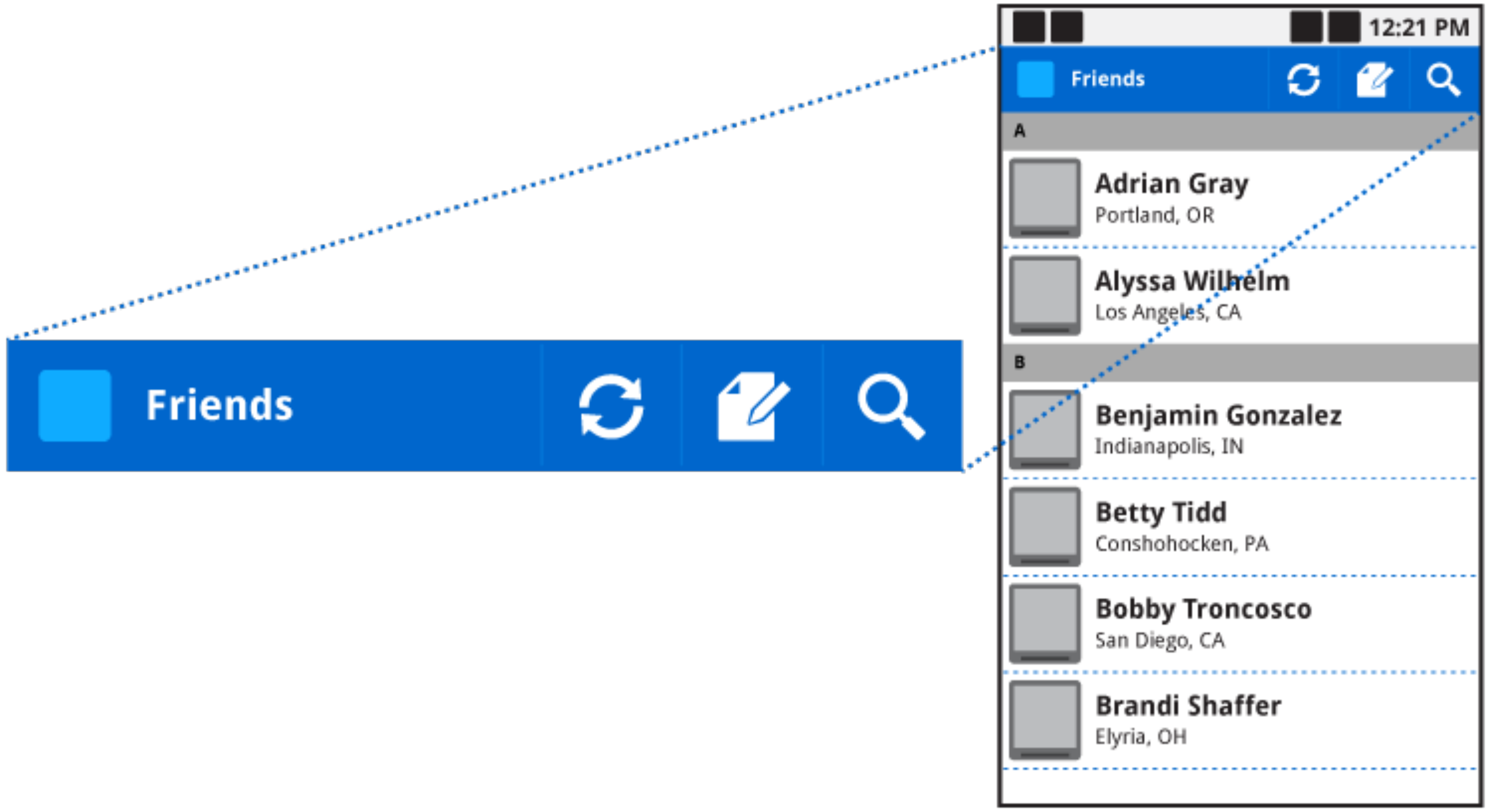
How did I wind up here?



How did I wind up here?



The Action Bar (alpha)

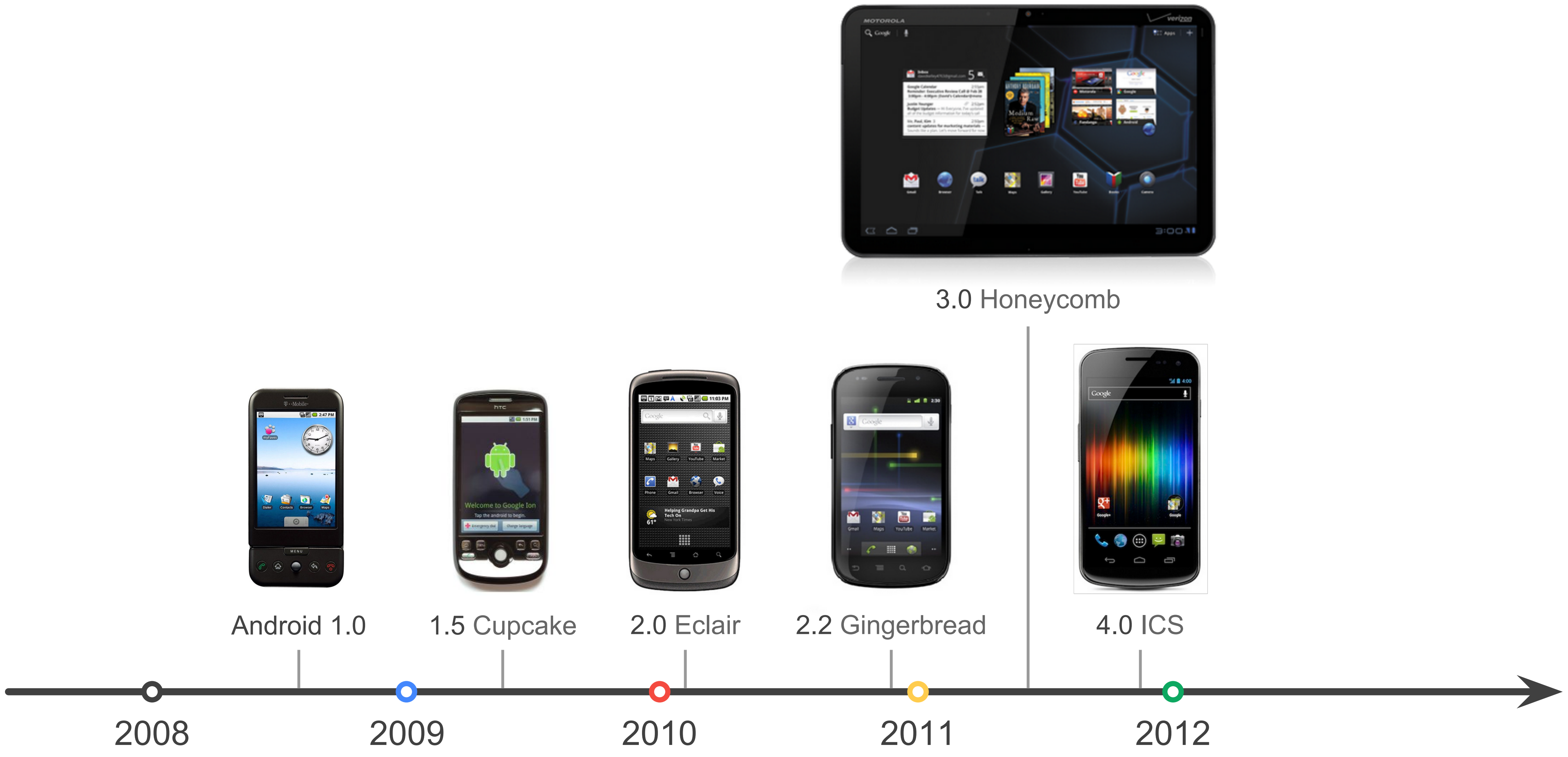


Limitations of the early Action Bar

But ...

- Not supported by the framework
- Only supports navigation directly to top of app
  - Less helpful for deeper apps

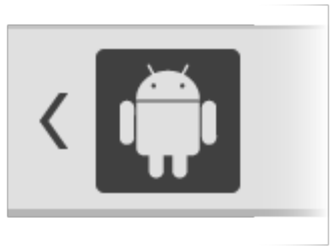
2011: Support for Navigation



Learning from Users

- Frequent, iterative user testing for Honeycomb
- Tested a variety of system and app navigation strategies
- Big insights:
    - Navigation in upper-left corner of the screen attractive for users
    - Clear distinction between controls located "inside" and "outside" the app
    - Users craved safe navigation, guaranteed to keep you "inside" the app
    - Quick, obvious task-switching was important

New navigation controls for Honeycomb and ICS

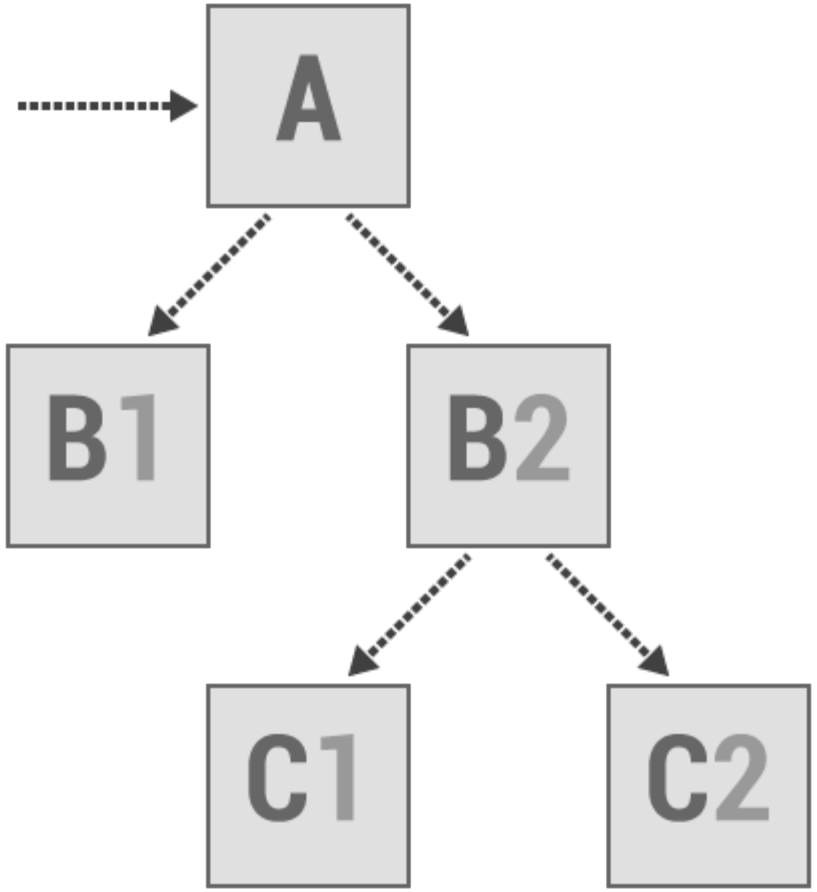


Up

Recents

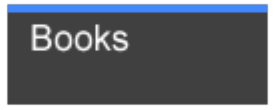
How Up works

- Structural navigation through your app
  - Guaranteed to keep you within the app
  - Available only for non-"home" activities
- Not dependent on state of the task stack
  - Based purely on the hierarchical relationships between activities
  - But, can alter the task stack as side effect
- **Up is not Back**
  - Often they may have the same effect
  - But not always...

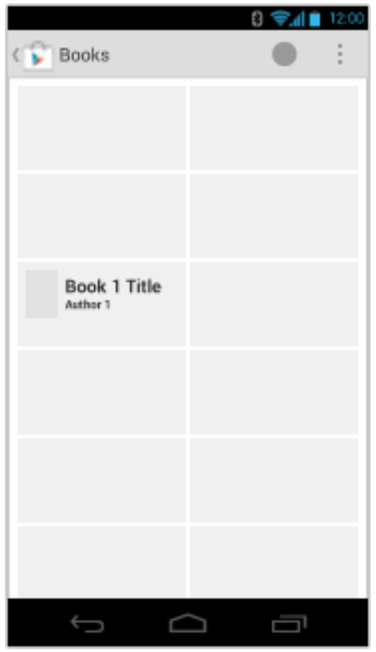


Up is Not Back: Related detail views

When linking to related content, Up can bypass intermediates in task stack



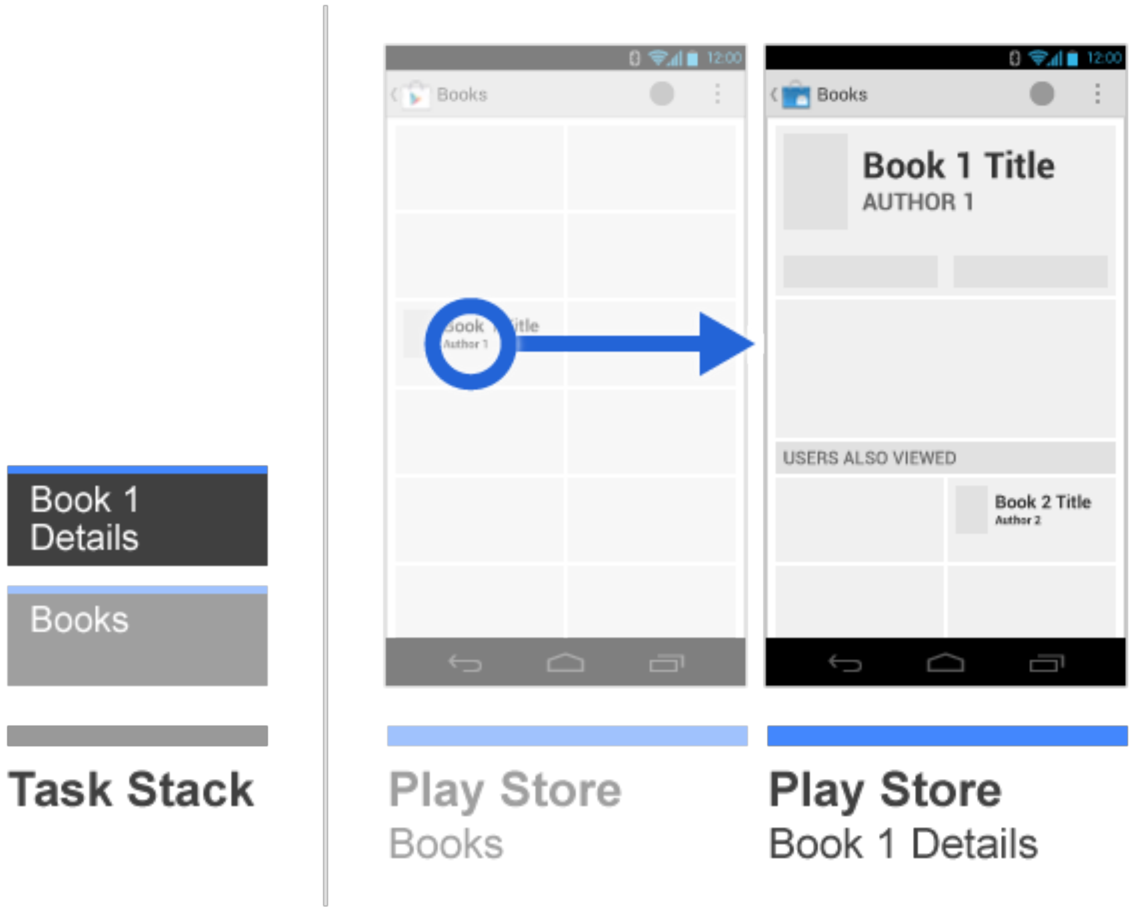
Task Stack



Play Store
Books

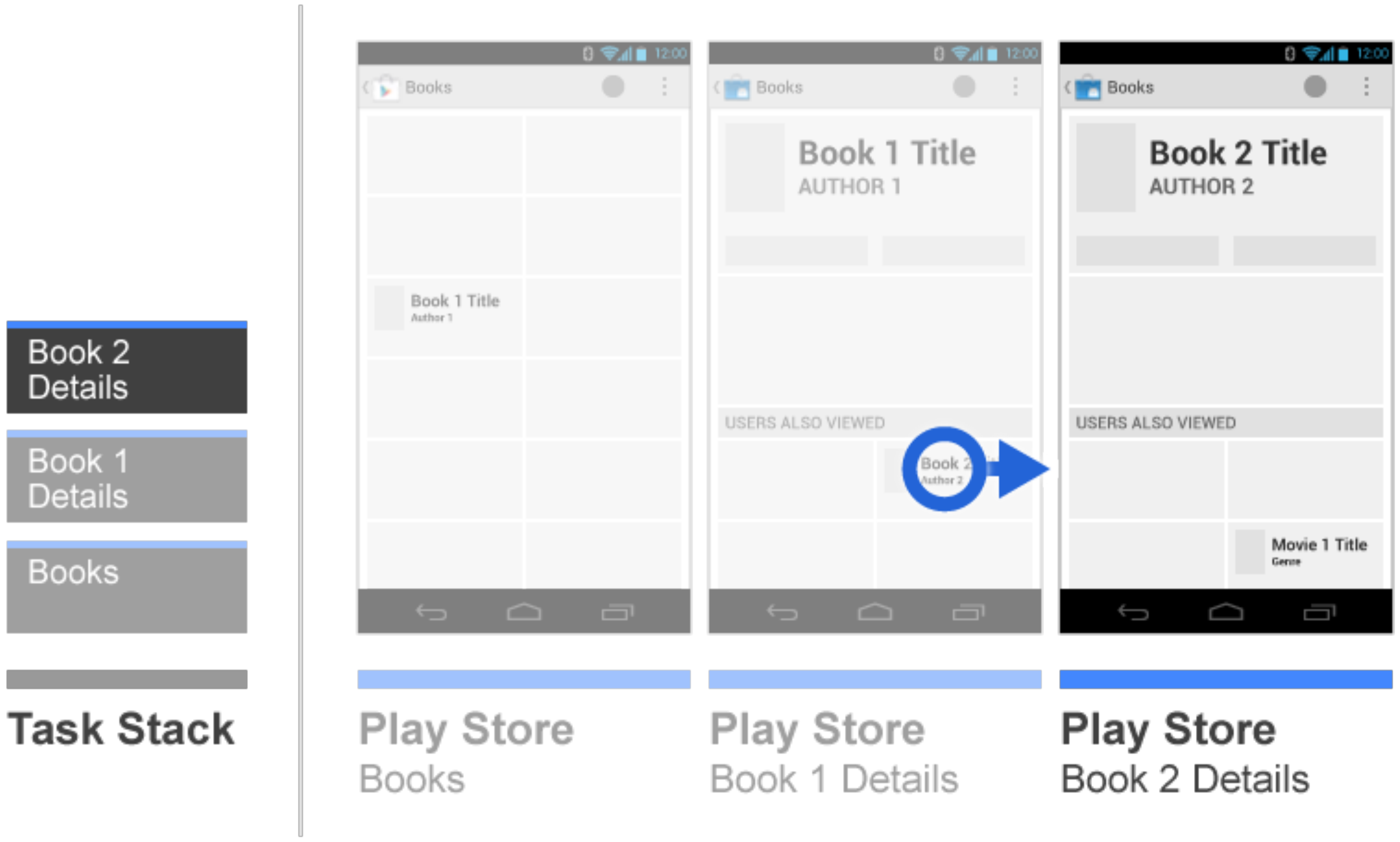
Up is Not Back: Related detail views

When linking to related content, Up can bypass intermediates in task stack



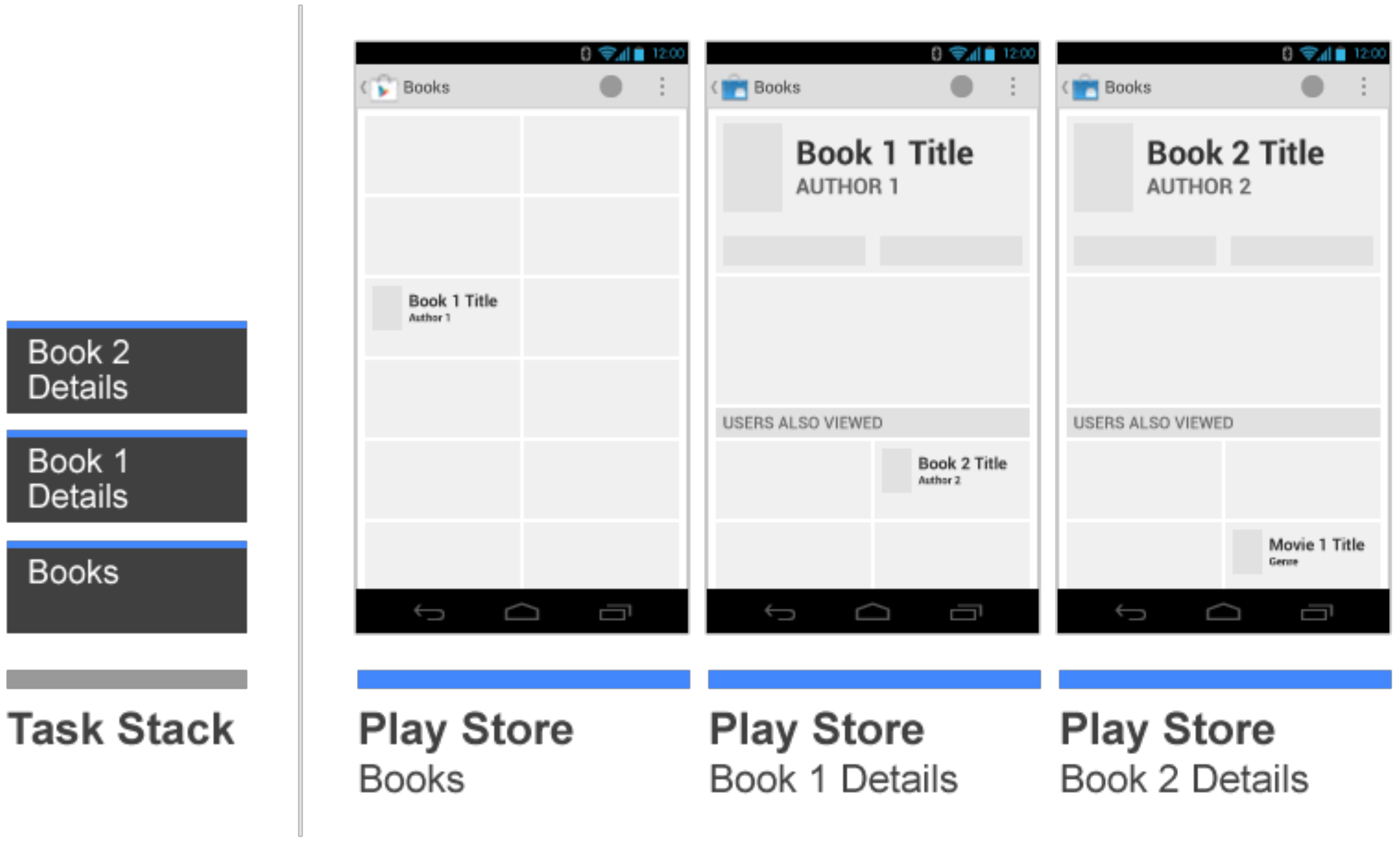
Up is Not Back: Related detail views

When linking to related content, Up can bypass intermediates in task stack



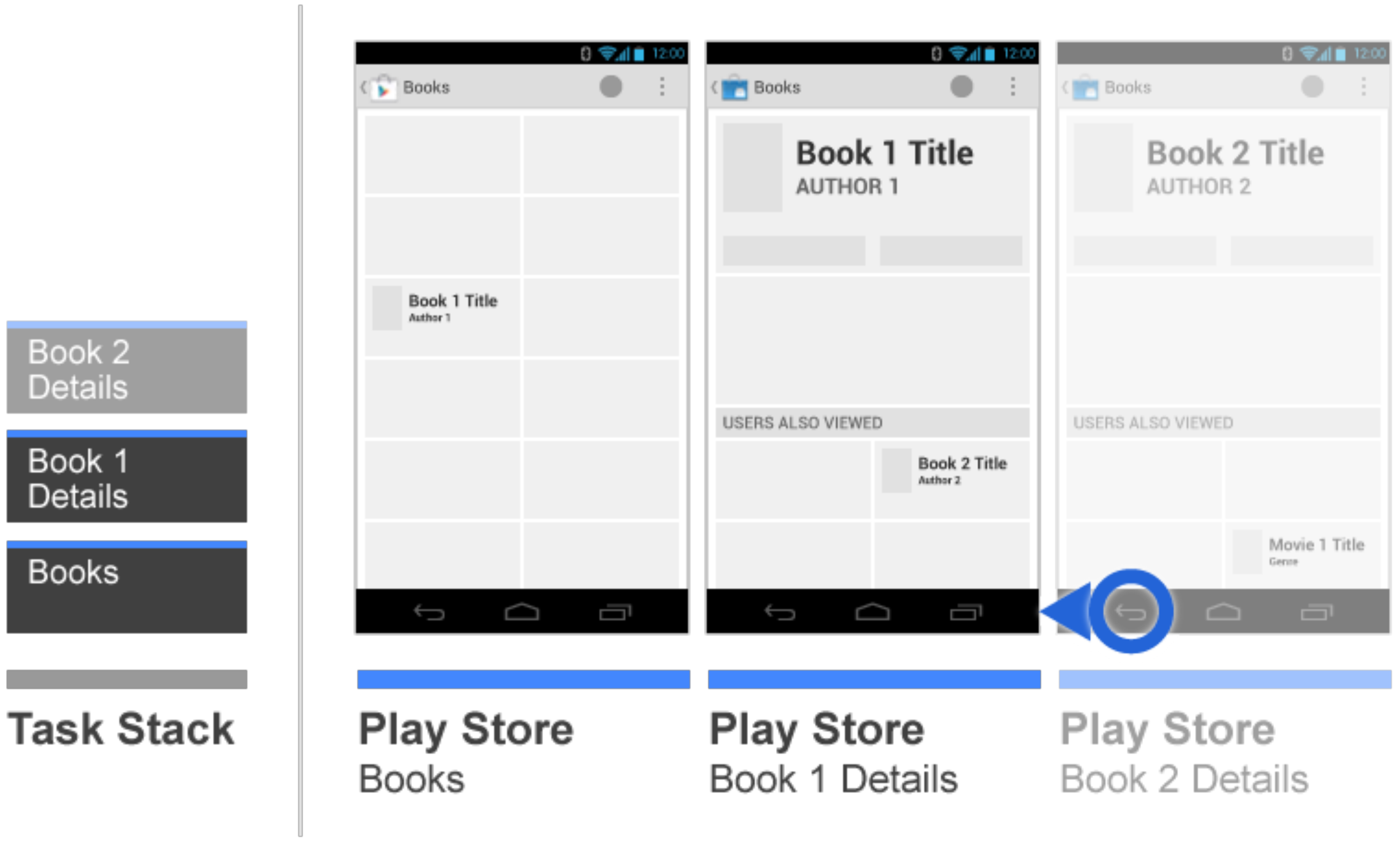
Up is Not Back: Related detail views

When linking to related content, Up can bypass intermediaries in task stack



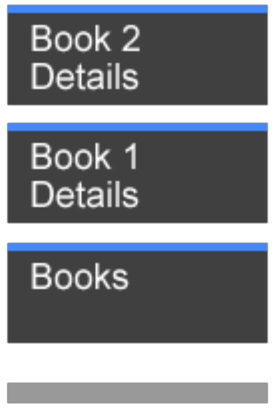
Up is Not Back: Related detail views

When linking to related content, Up can bypass intermediates in task stack

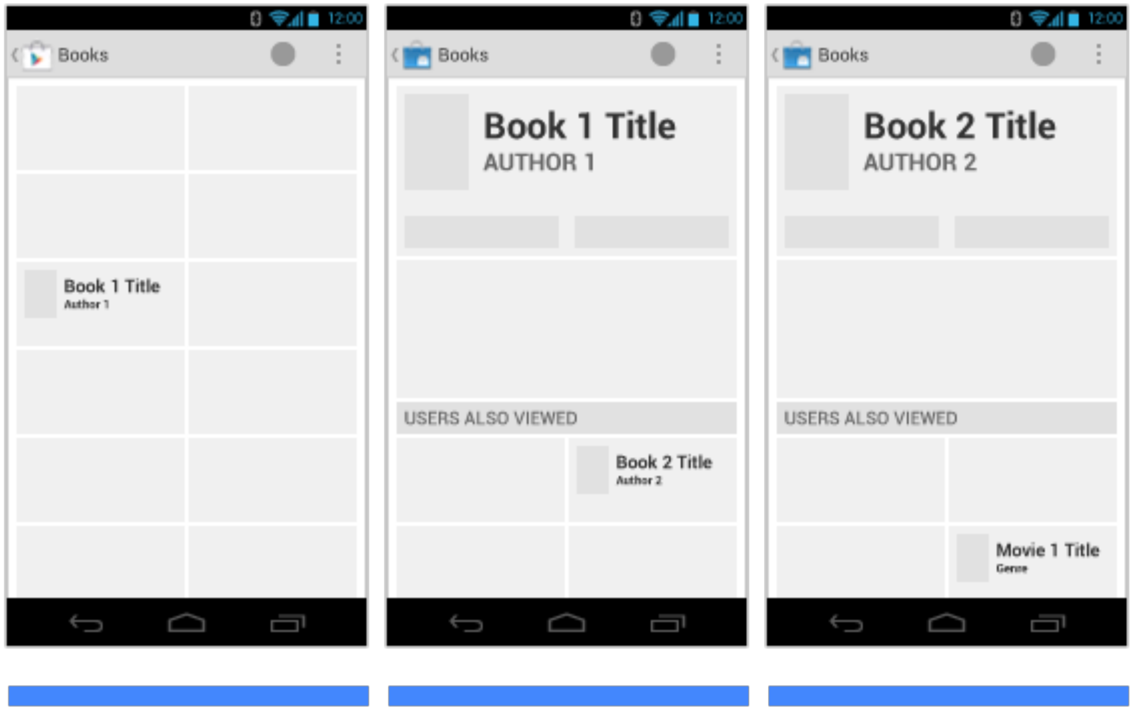


Up is Not Back: Related detail views

When linking to related content, Up can bypass intermediates in task stack



Task Stack



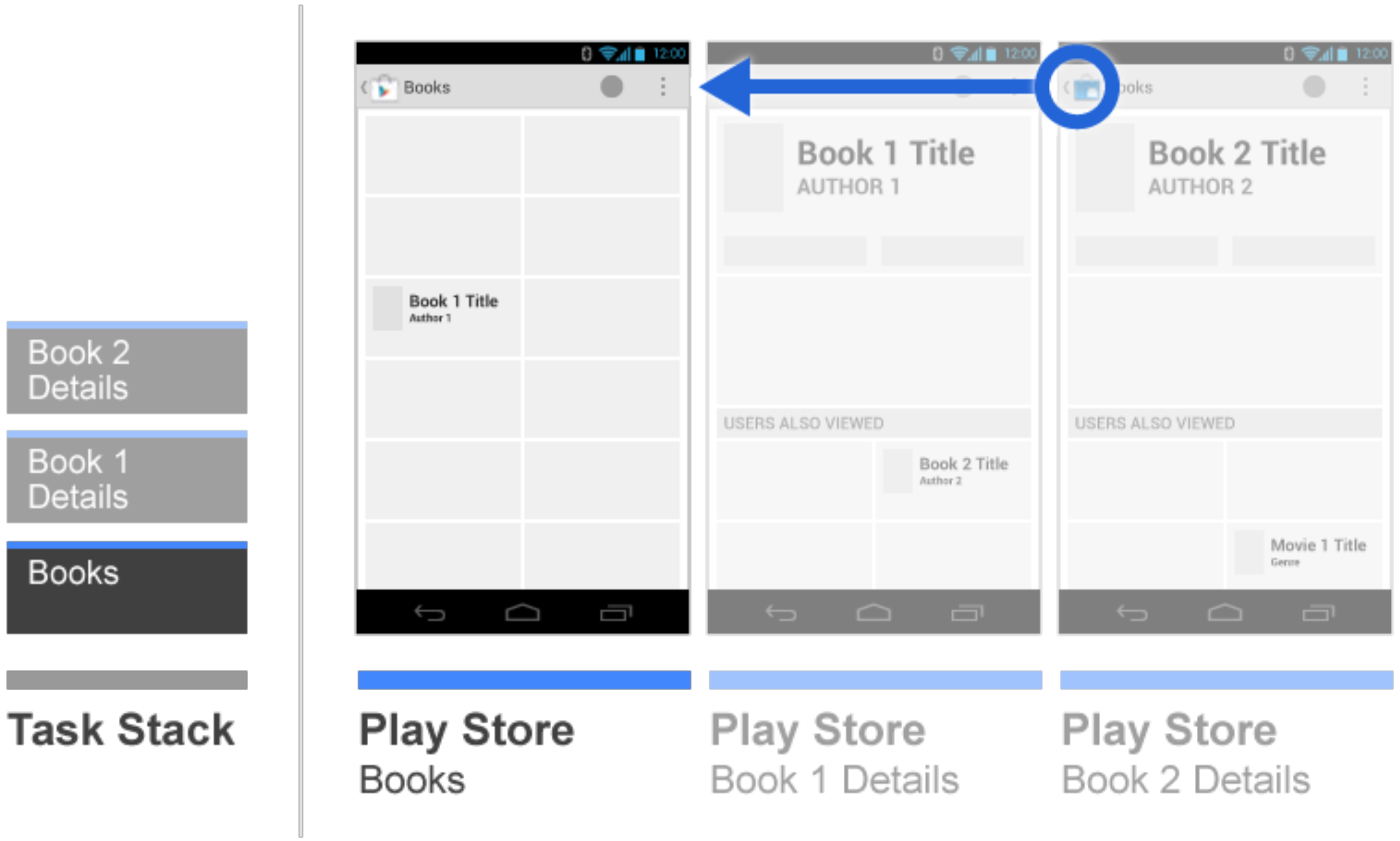
Play Store
Books

Play Store
Book 1 Details

Play Store
Book 2 Details

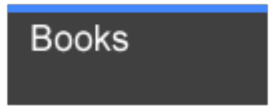
Up is Not Back: Related detail views

When linking to related content, Up can bypass intermediates in task stack

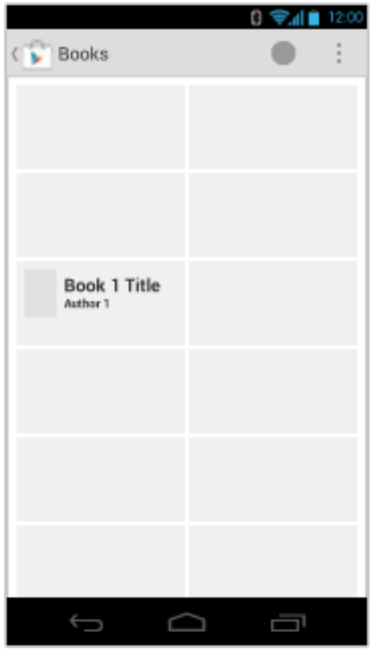


Up is Not Back: Related detail views

When linking to related content, Up can bypass intermediates in task stack



Task Stack



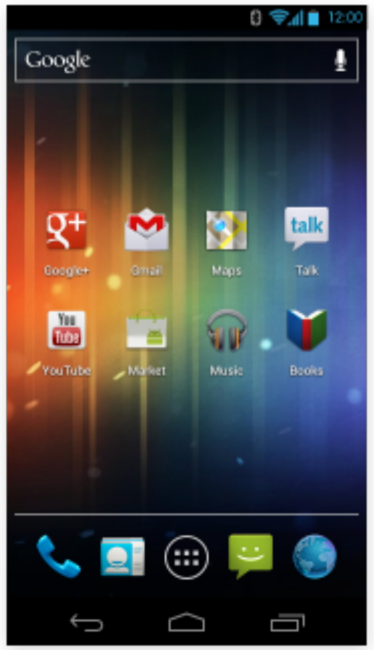
Play Store
Books

Up is Not Back: Tasks built from multiple apps

If activities from multiple apps are present in a task, Up can create a new task

Launcher

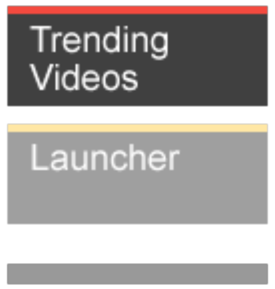
Task Stack



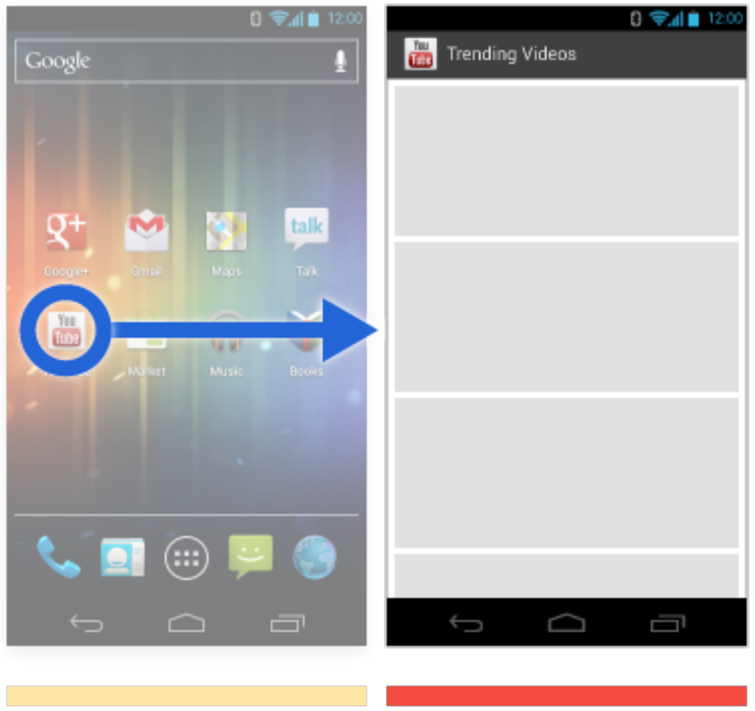
Home
Launcher

Up is Not Back: Tasks built from multiple apps

If activities from multiple apps are present in a task, Up can create a new task



Task Stack

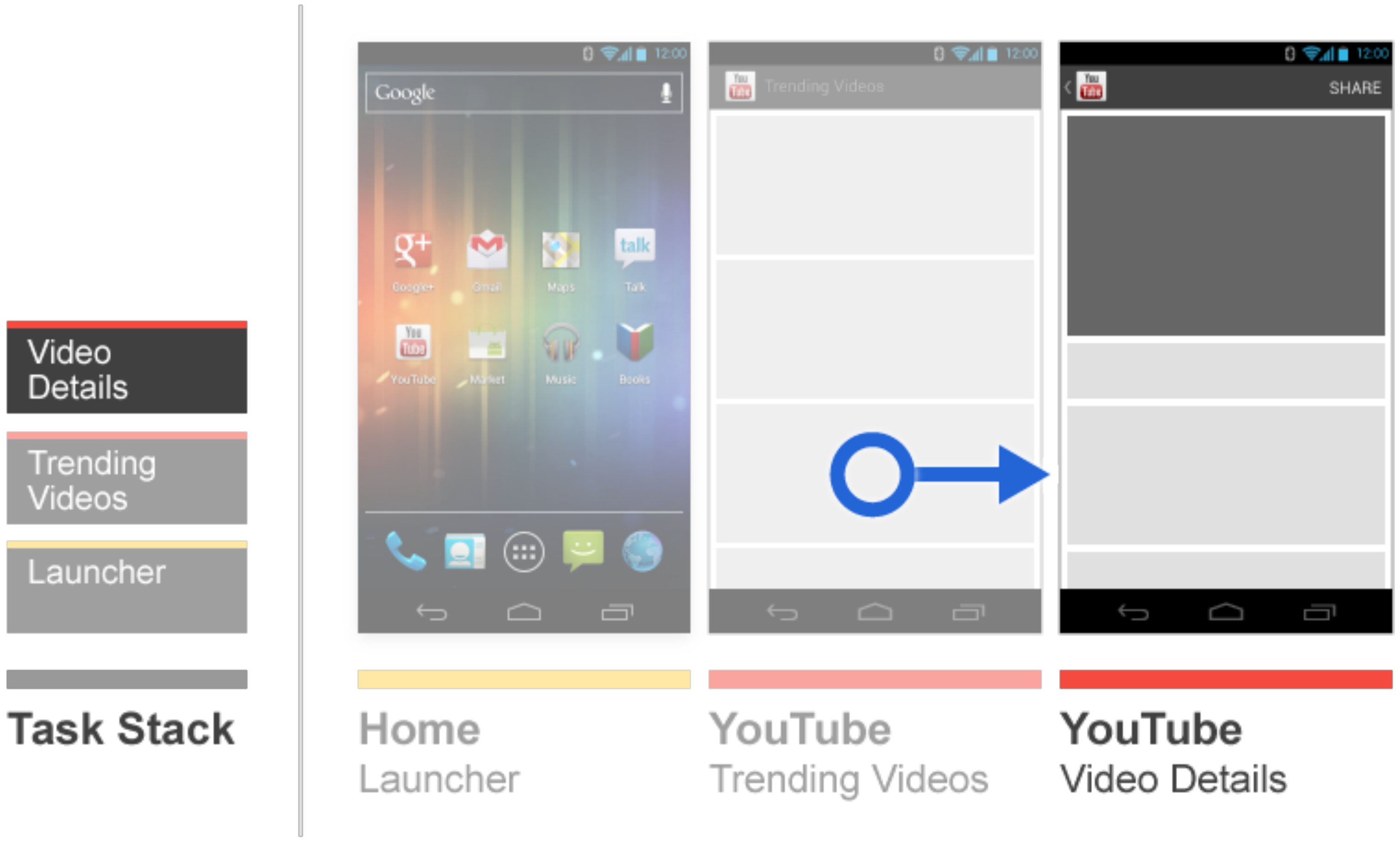


Home
Launcher

YouTube
Trending Videos

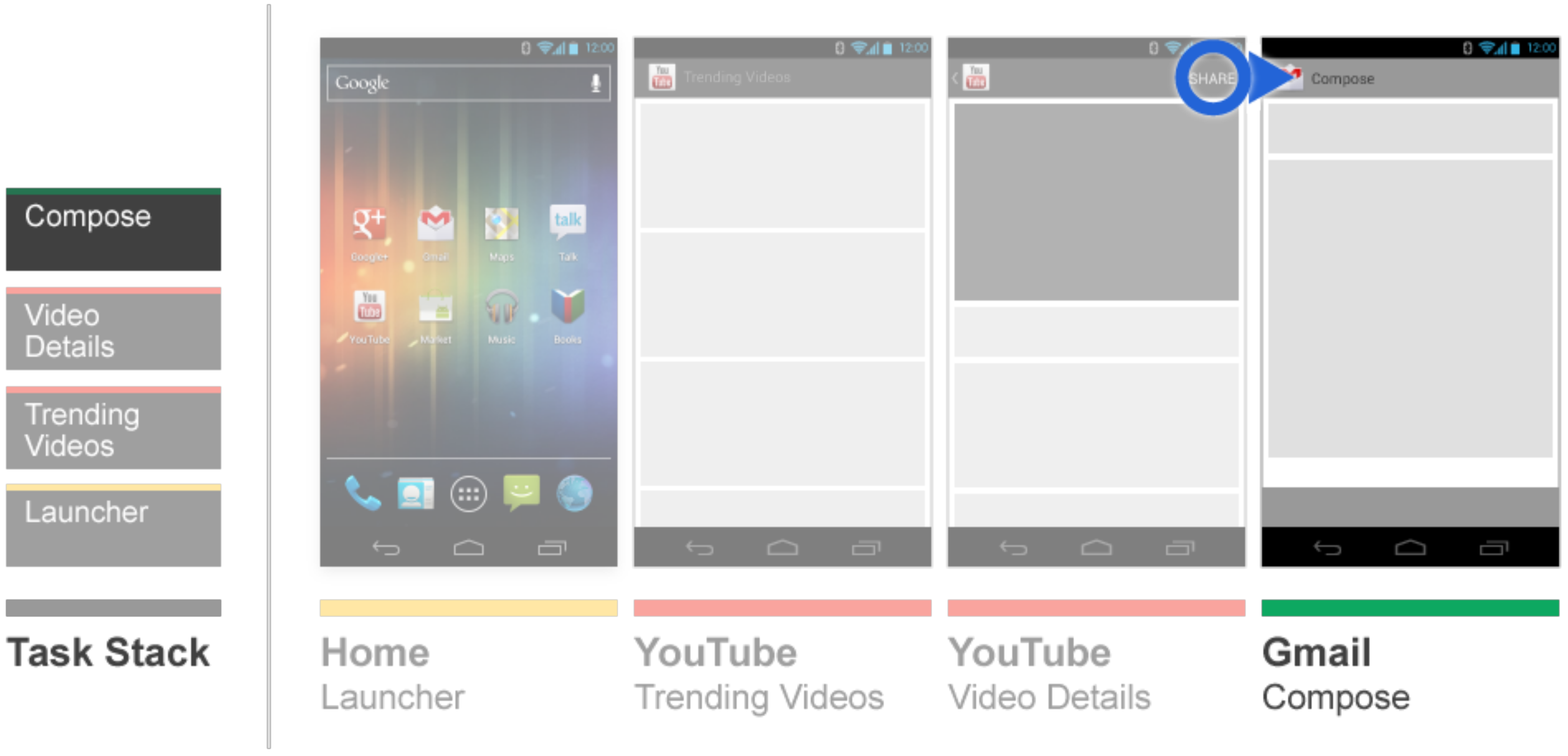
Up is Not Back: Tasks built from multiple apps

If activities from multiple apps are present in a task, Up can create a new task



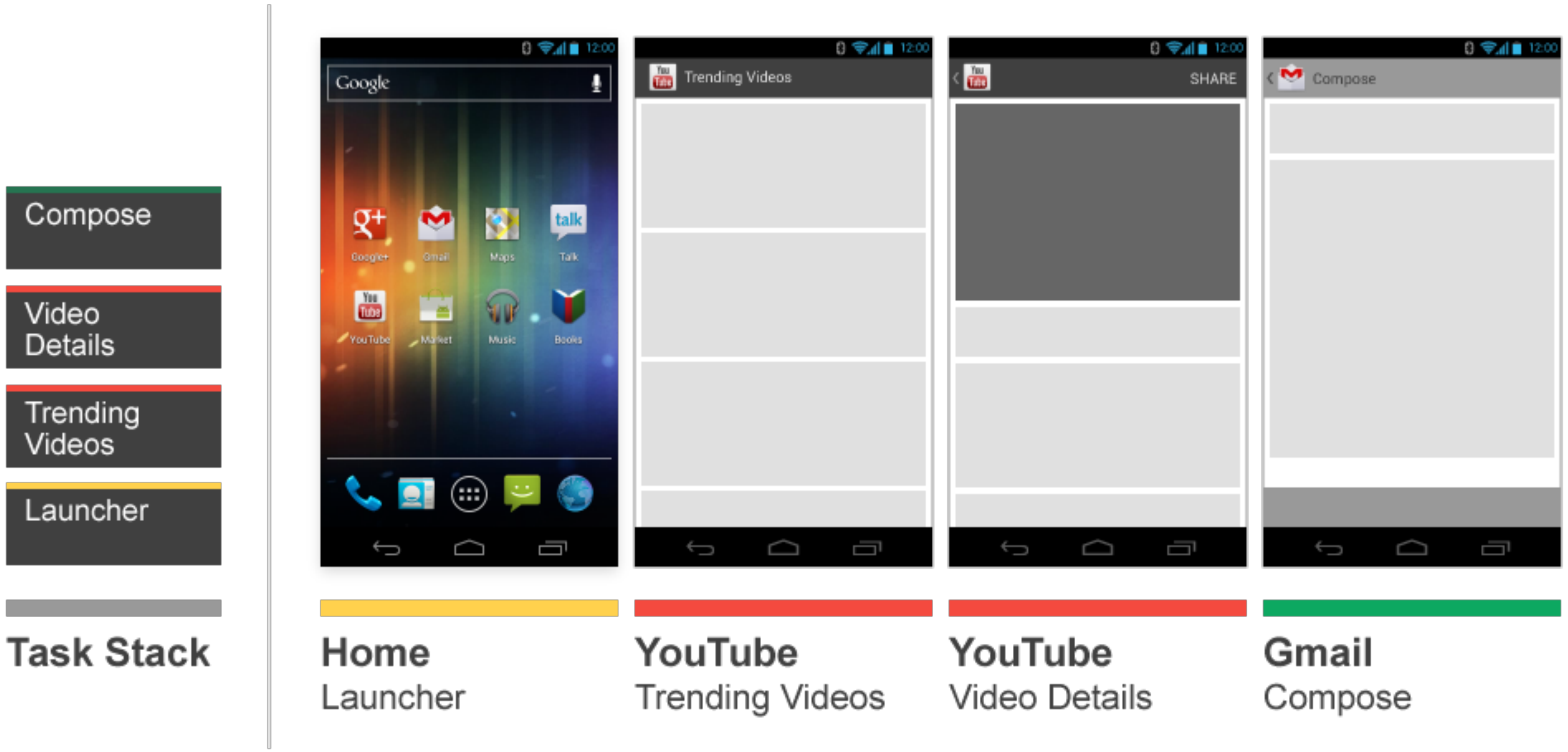
Up is Not Back: Tasks built from multiple apps

If activities from multiple apps are present in a task, Up can create a new task



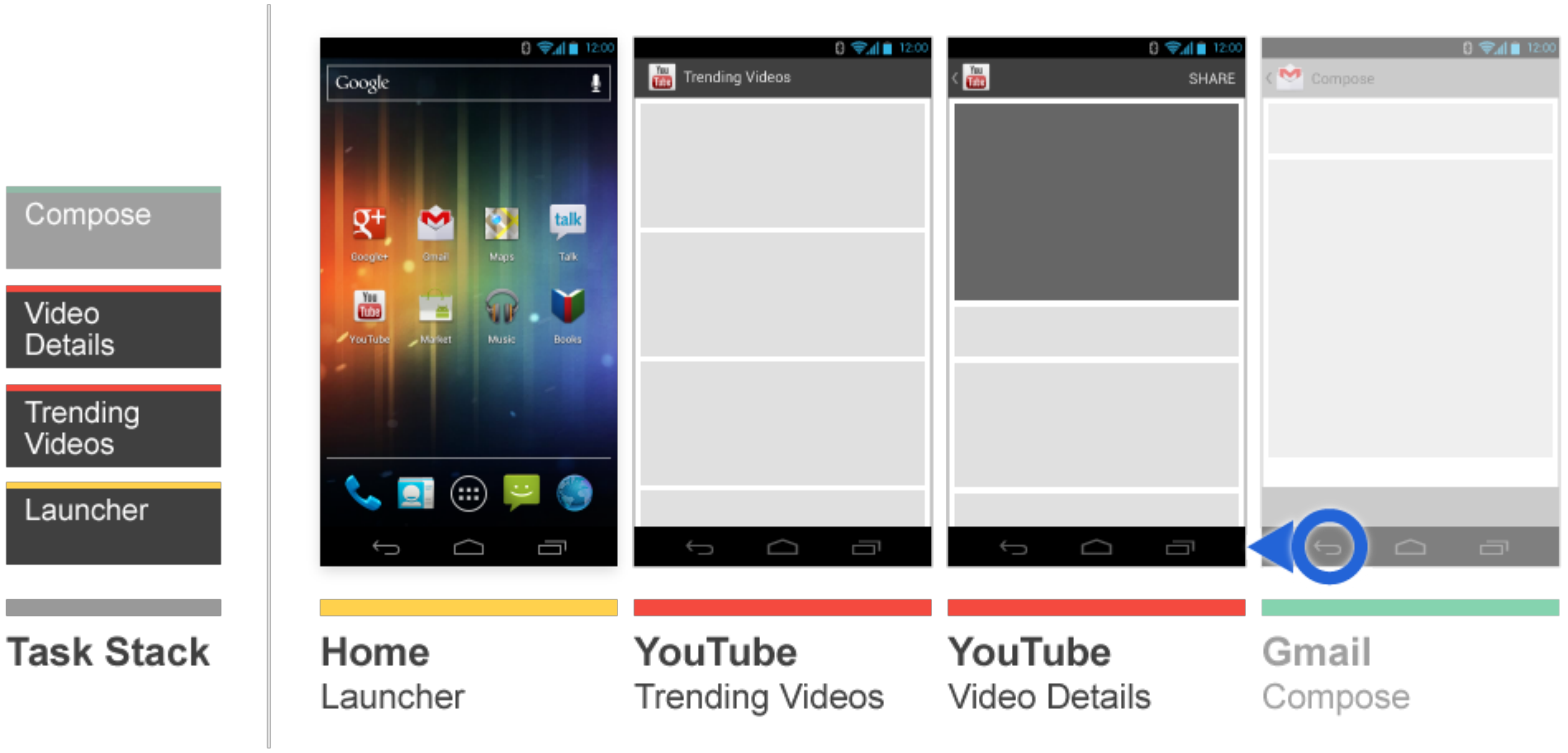
Up is Not Back: Tasks built from multiple apps

If activities from multiple apps are present in a task, Up can create a new task



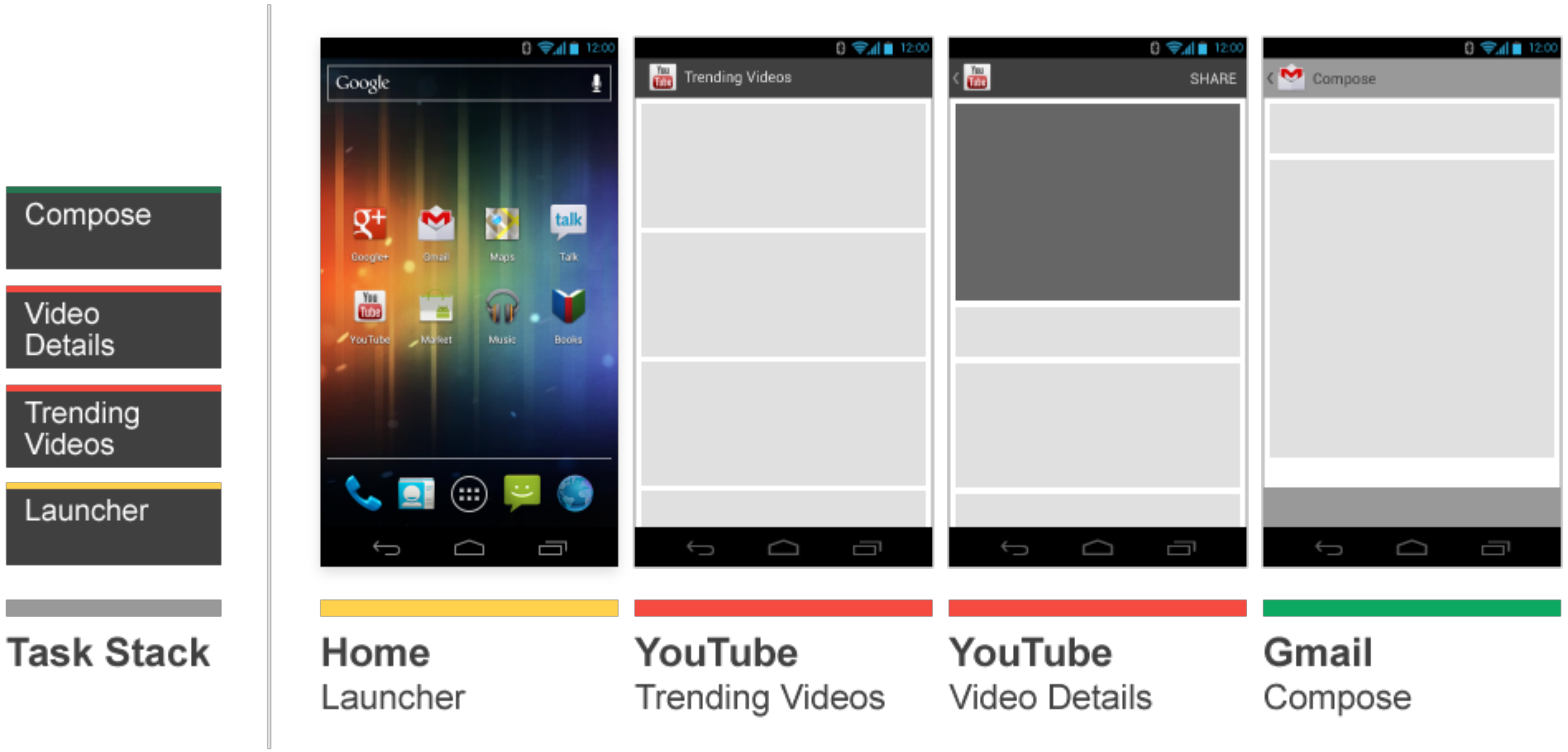
Up is Not Back: Tasks built from multiple apps

If activities from multiple apps are present in a task, Up can create a new task



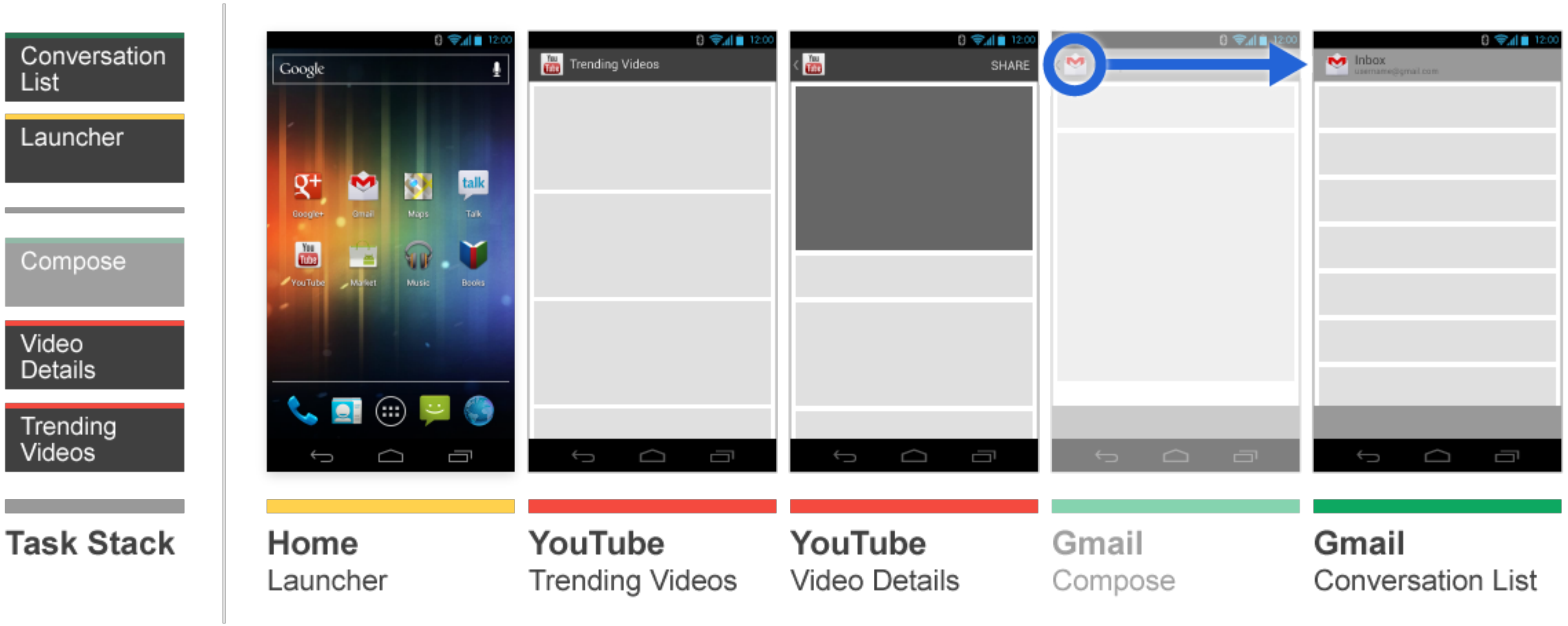
Up is Not Back: Tasks built from multiple apps

If activities from multiple apps are present in a task, Up can create a new task

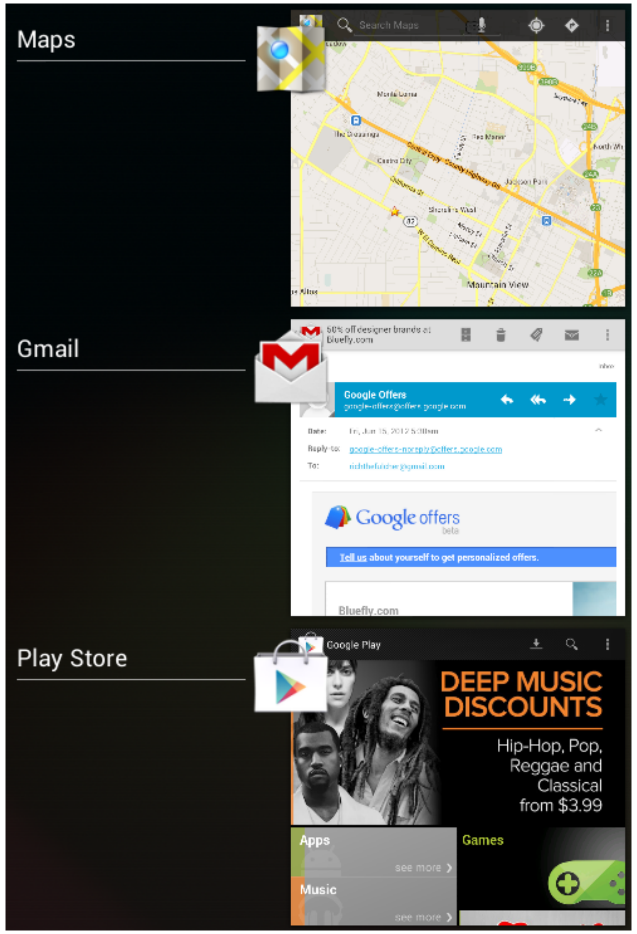


Up is Not Back: Tasks built from multiple apps

If activities from multiple apps are present in a task, Up can create a new task



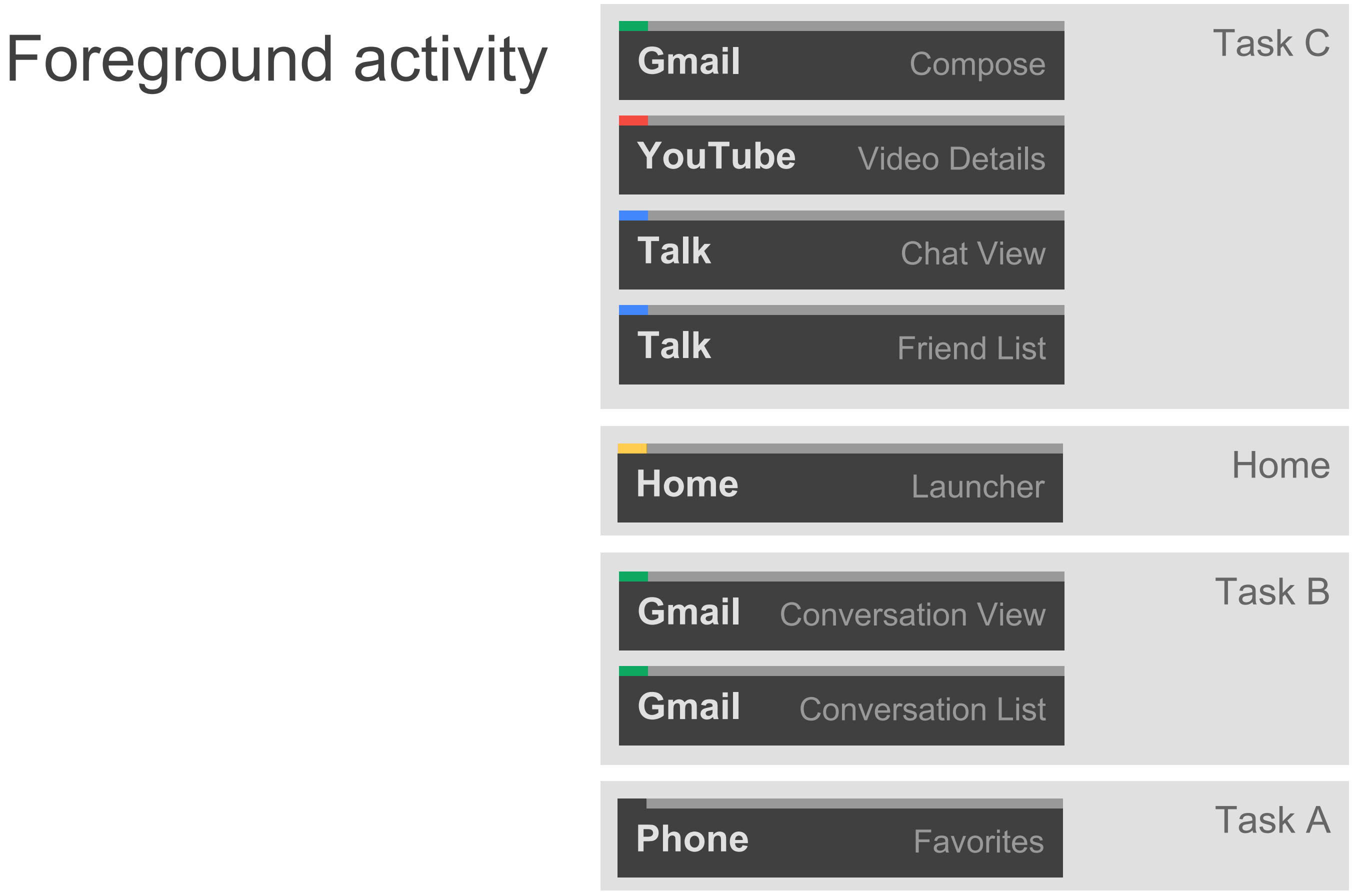
How Recents works



- Navigation between tasks
- Brings entire task forward on global activity stack
- Easily accessible from device's navigation bar

How ActivityManager sees the world

Bringing Task B to the top

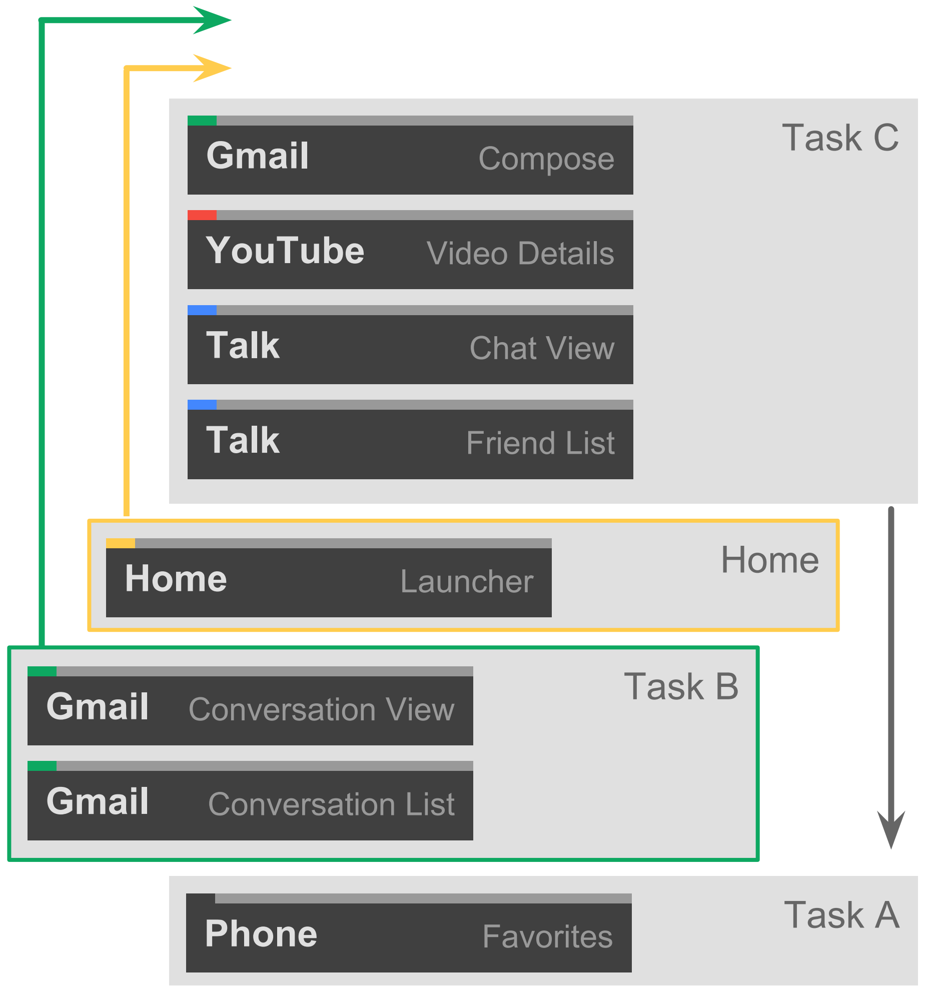


How ActivityManager sees the world

Bringing Task B to the top

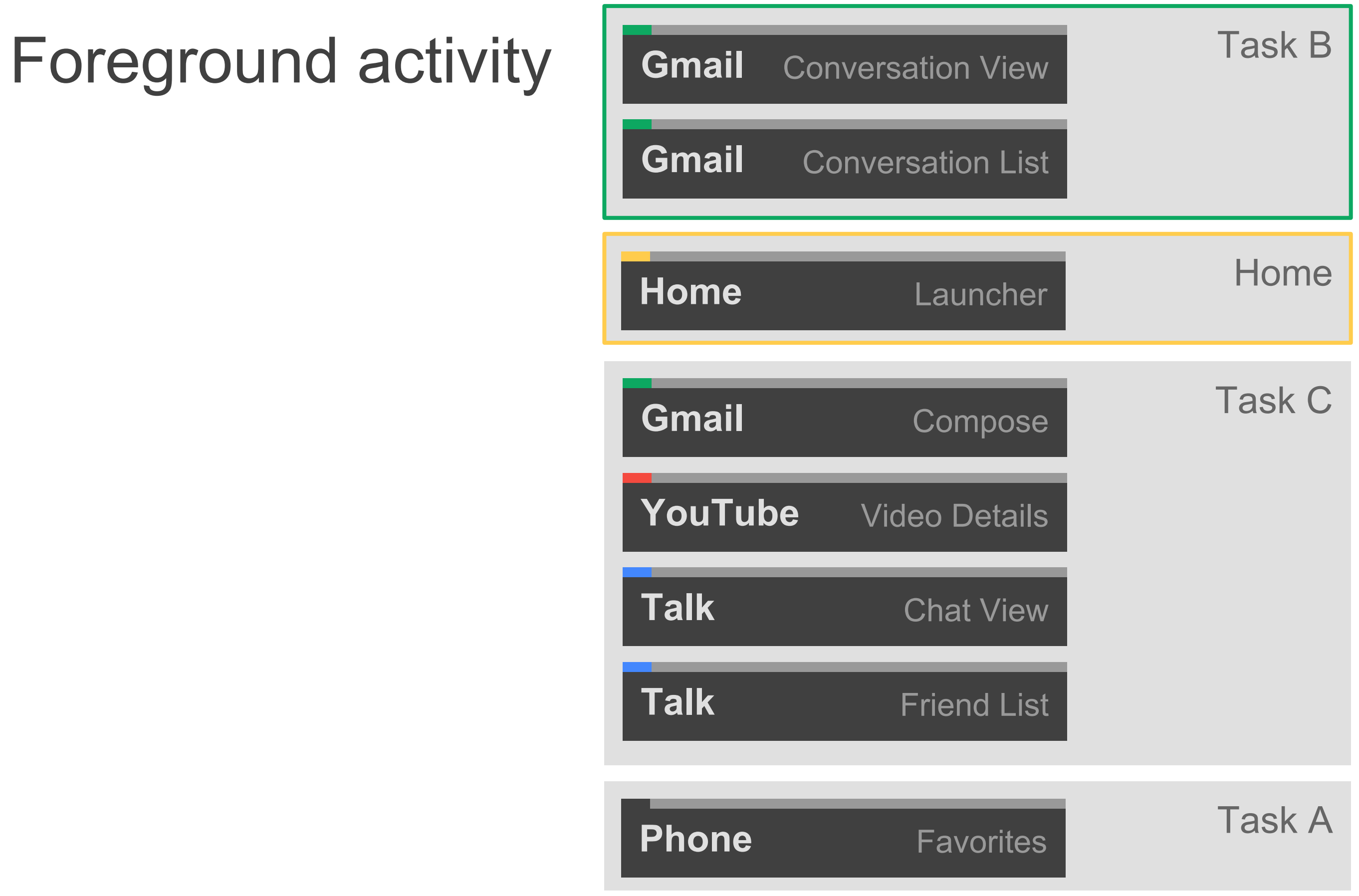
`Intent.FLAG_ACTIVITY_NEW_TASK`

`Intent.FLAG_ACTIVITY_TASK_ON_HOME`

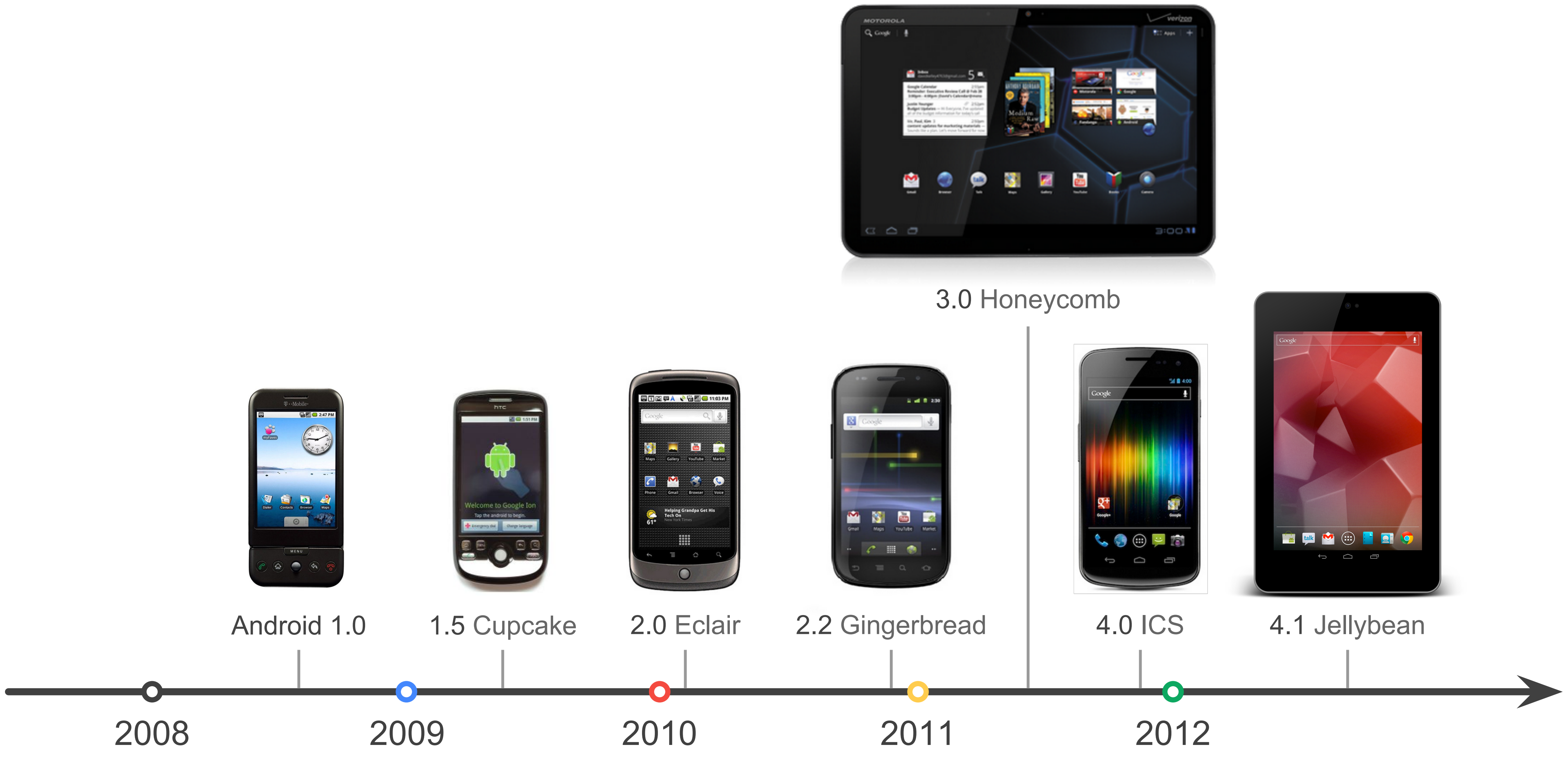


How ActivityManager sees the world

Bringing Task B to the top



And today, we announce ... nothing else new!



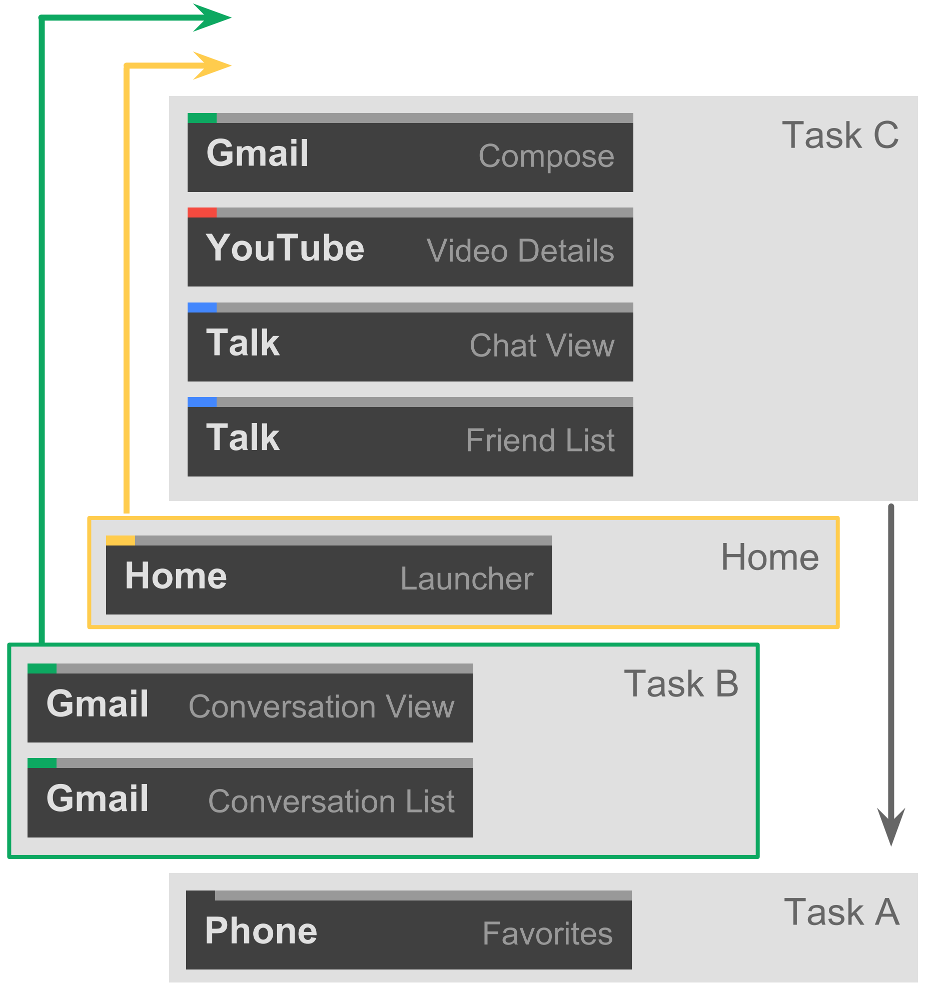
Now that you know the history...

- It's easy to find local maxima for your app's experience
  - Overload Back; use highly custom navigation structures
- Consistency makes the whole system feel intuitive and natural
  - You may know better for *your* app...
- We can't do it without your help!
- But how?

Switching tasks more cleanly (Android 3.0+)

Bring Launcher along with Intent.FLAG_ACTIVITY_TASK_ON_HOME

```
Intent.FLAG_ACTIVITY_NEW_TASK

Intent.FLAG_ACTIVITY_TASK_ON_HOME
```



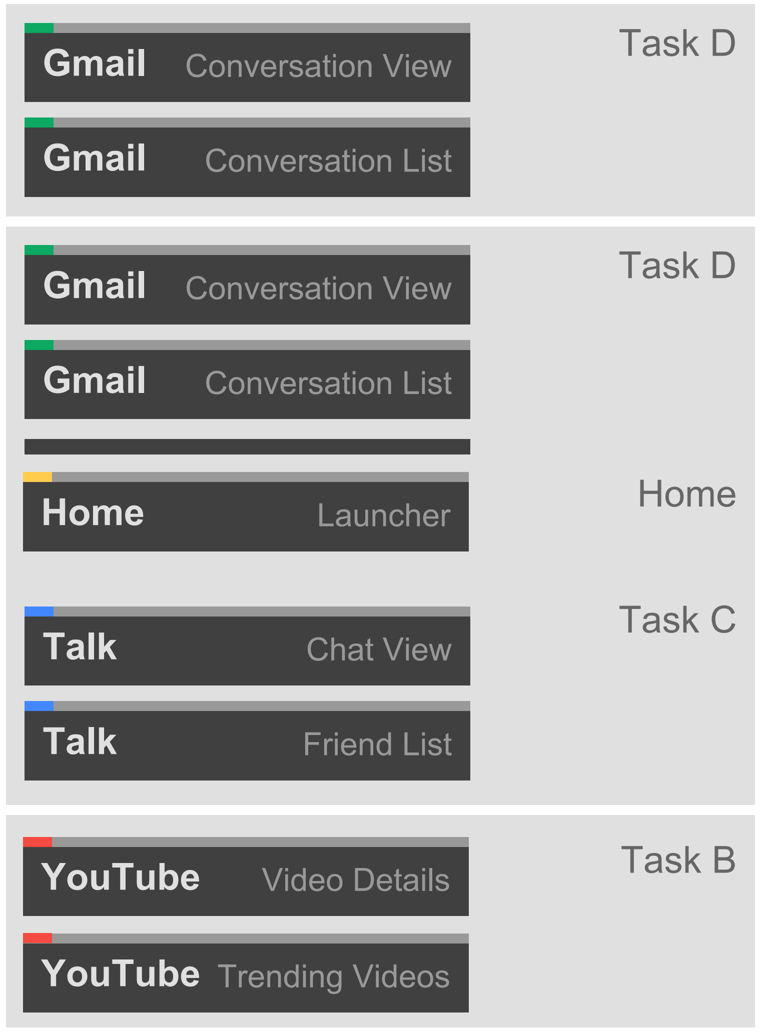
Why?
Stability and predictability over time.

Faking History

- Why would you replace a task?
- Starting a new task with an Activity down the app's hierarchy
    - Example: Up, Widgets
- Provides stable navigation when returning to that task later
    - User doesn't need to remember how they got there

Fully replace an existing task (Android 3.0+)

Use `startActivities(Intent[])`
and `Intent.FLAG_ACTIVITY_CLEAR_TASK`



Making it easy: TaskStackBuilder

- Available in Jellybean (and the support library since earlier this year)
- Automatically builds a synthetic task stack from manifest metadata
- Pulls together `startActivities(Intent[])`, `FLAG_ACTIVITY_NEW_TASK`, `FLAG_ACTIVITY_CLEAR_TASK`, `FLAG_ACTIVITY_TASK_ON_HOME`
- Also makes `PendingIntents` for notifications and launcher widgets
- See the AppNavigation/SupportAppNavigation samples in the SDK

Implementing Up

- Done for you in Jellybean!
  - Just give us the metadata in your manifest
  - Automatically detects the current task affinity and does the right thing

- Need to customize some Intent extras or other arguments?
  - No problem.

- Coming soon in the AppCompat lib
  - NavUtils and TaskStackBuilder are the building blocks

Widgets and Notifications

Deep Links

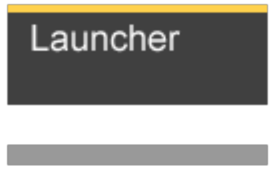
- Widgets and notifications often target activities deep within an app's hierarchy

- Widgets clearly initiate new tasks
    - They launch from Home, after all

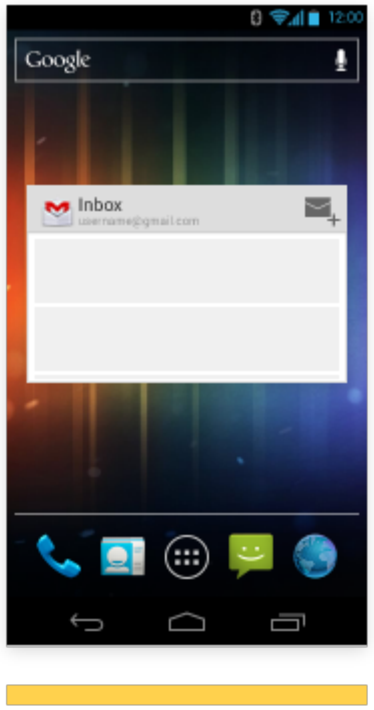
- Notifications are ... a bit trickier

Navigating via Widgets

Including unseen activities in the task stack



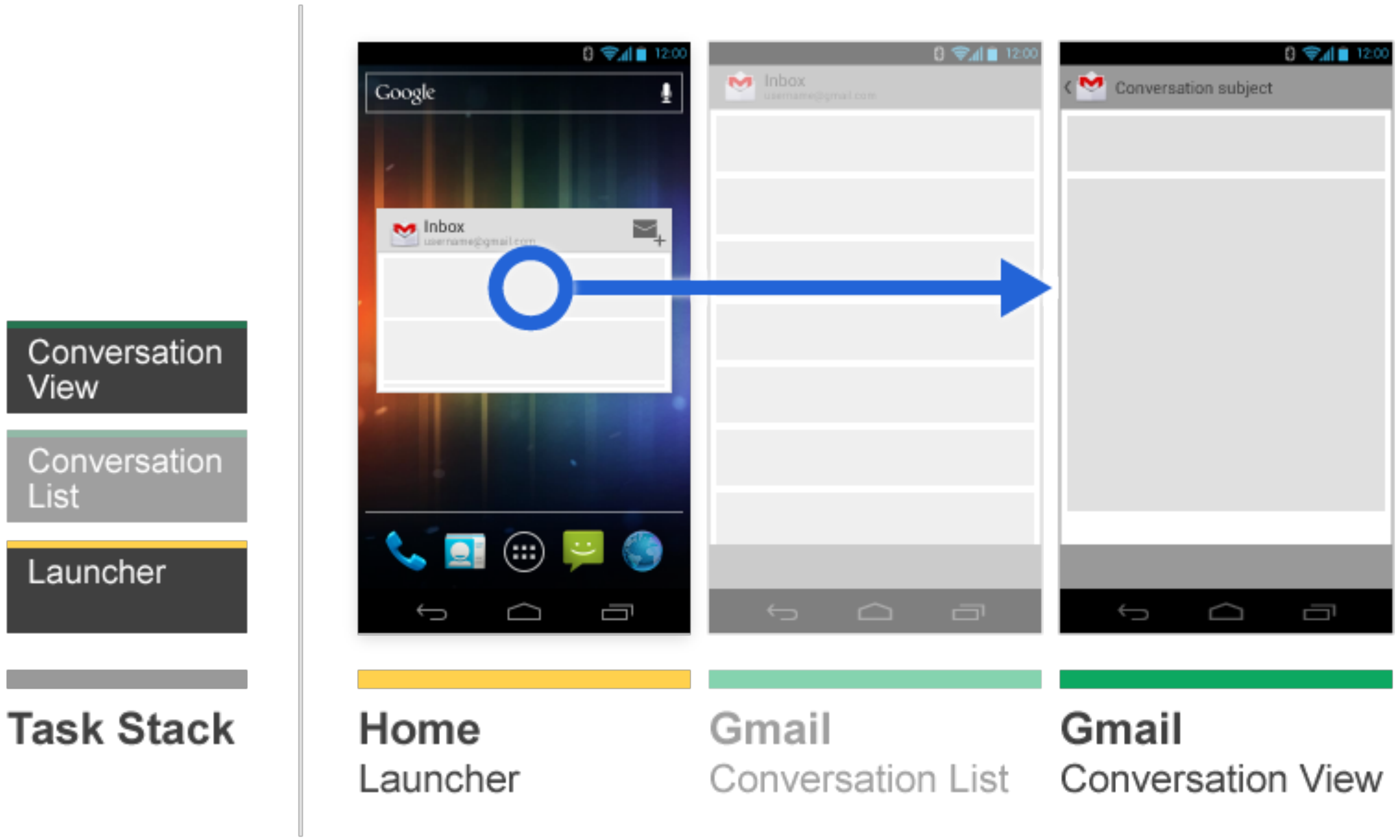
Task Stack



Home
Launcher

Navigating via Widgets

Including unseen activities in the task stack



Navigating via Widgets

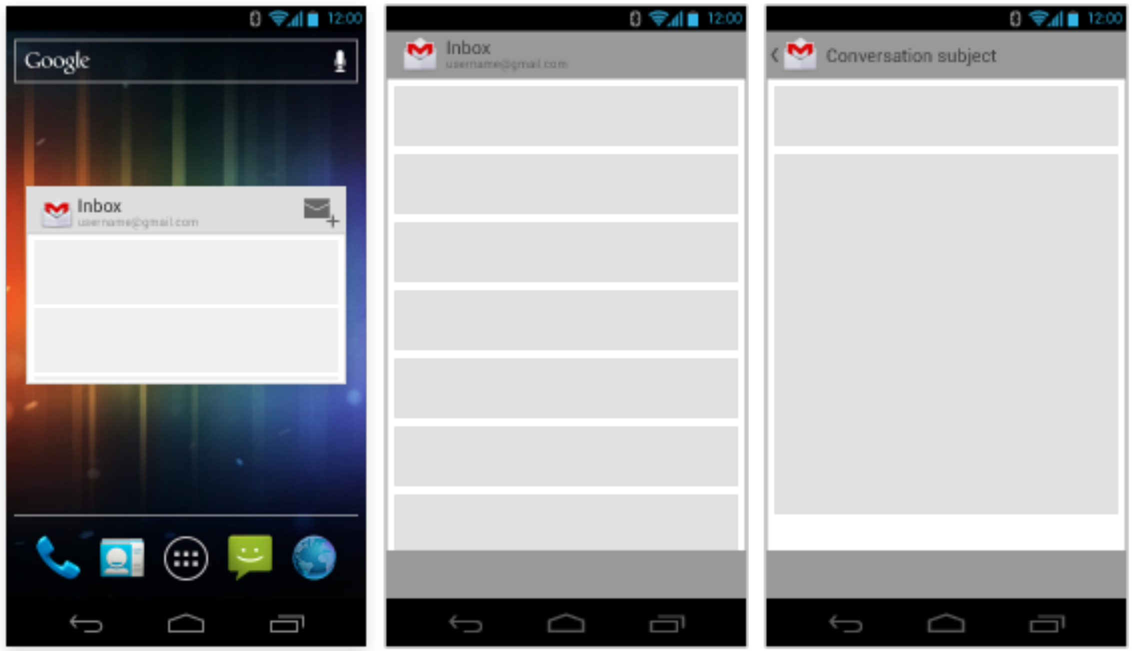
Including unseen activities in the task stack

Conversation View

Conversation List

Launcher

Task Stack



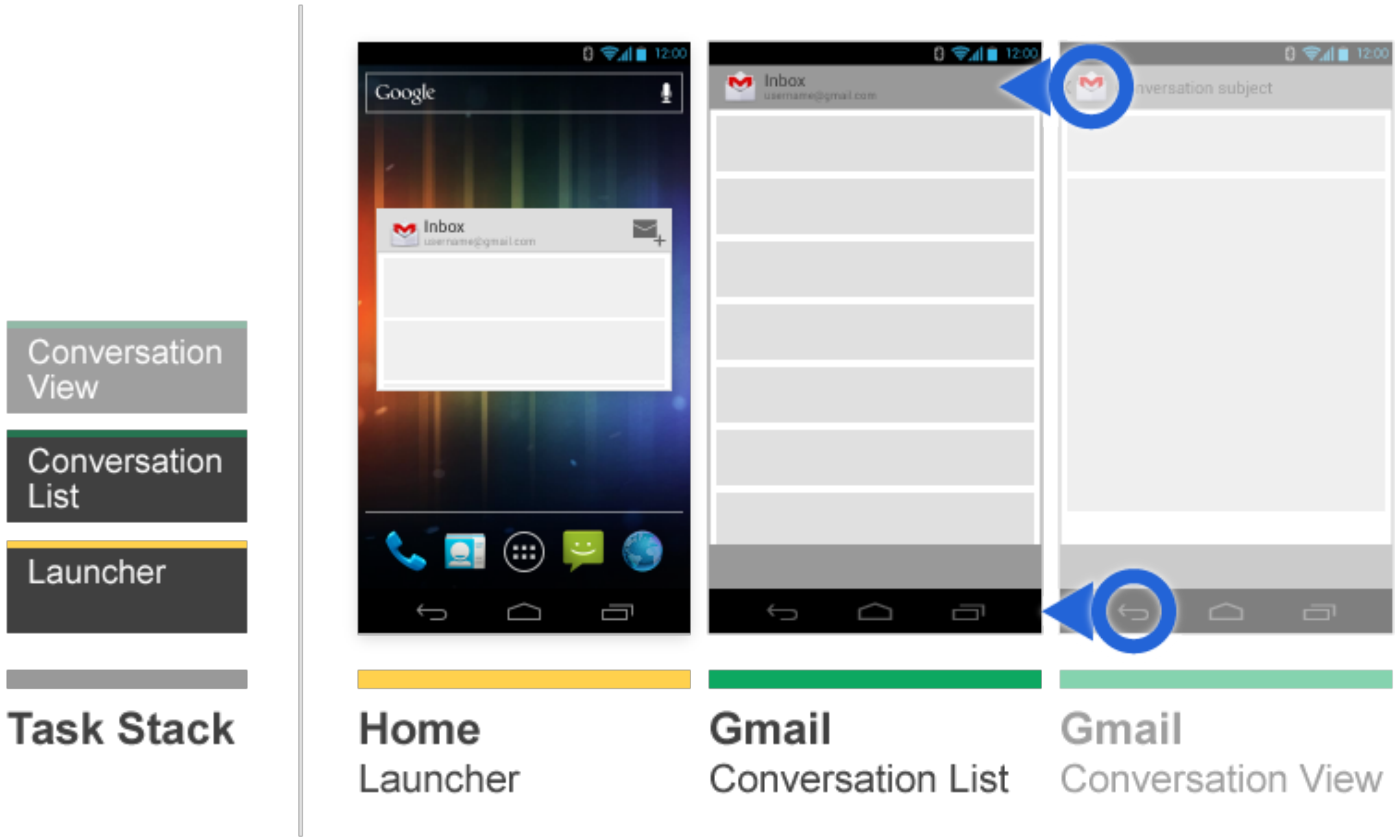
Home
Launcher

Gmail
Conversation List

Gmail
Conversation View

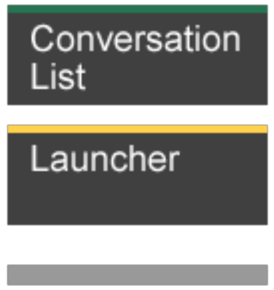
Navigating via Widgets

Including unseen activities in the task stack

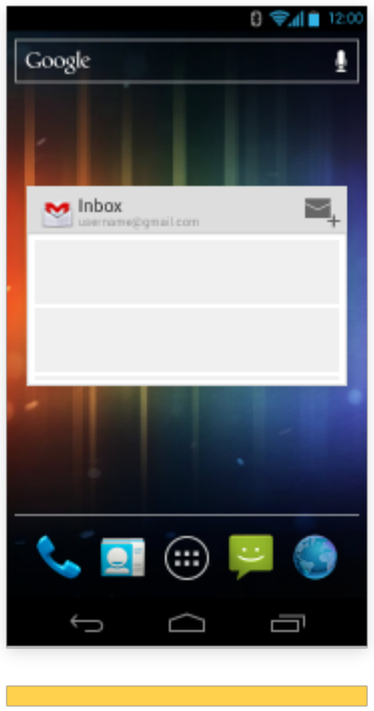


Navigating via Widgets

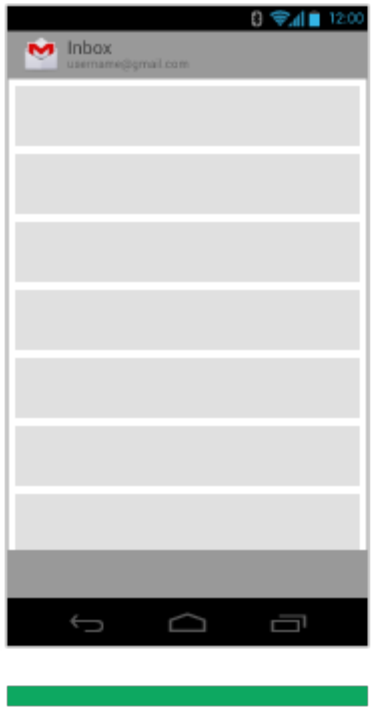
Including unseen activities in the task stack



Task Stack



Home
Launcher



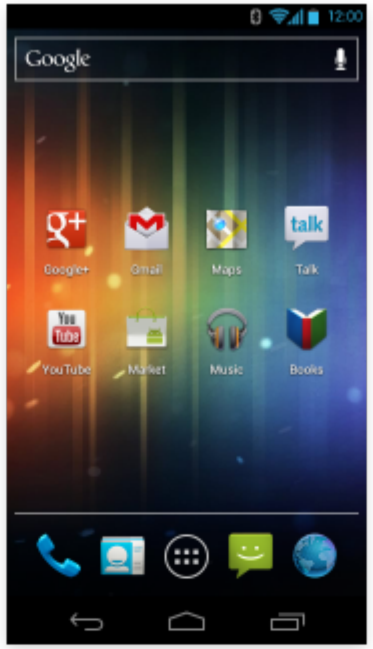
Gmail
Conversation List

The elephant in the room: Notifications

A common use case: follow notification directly from one app to another

Launcher

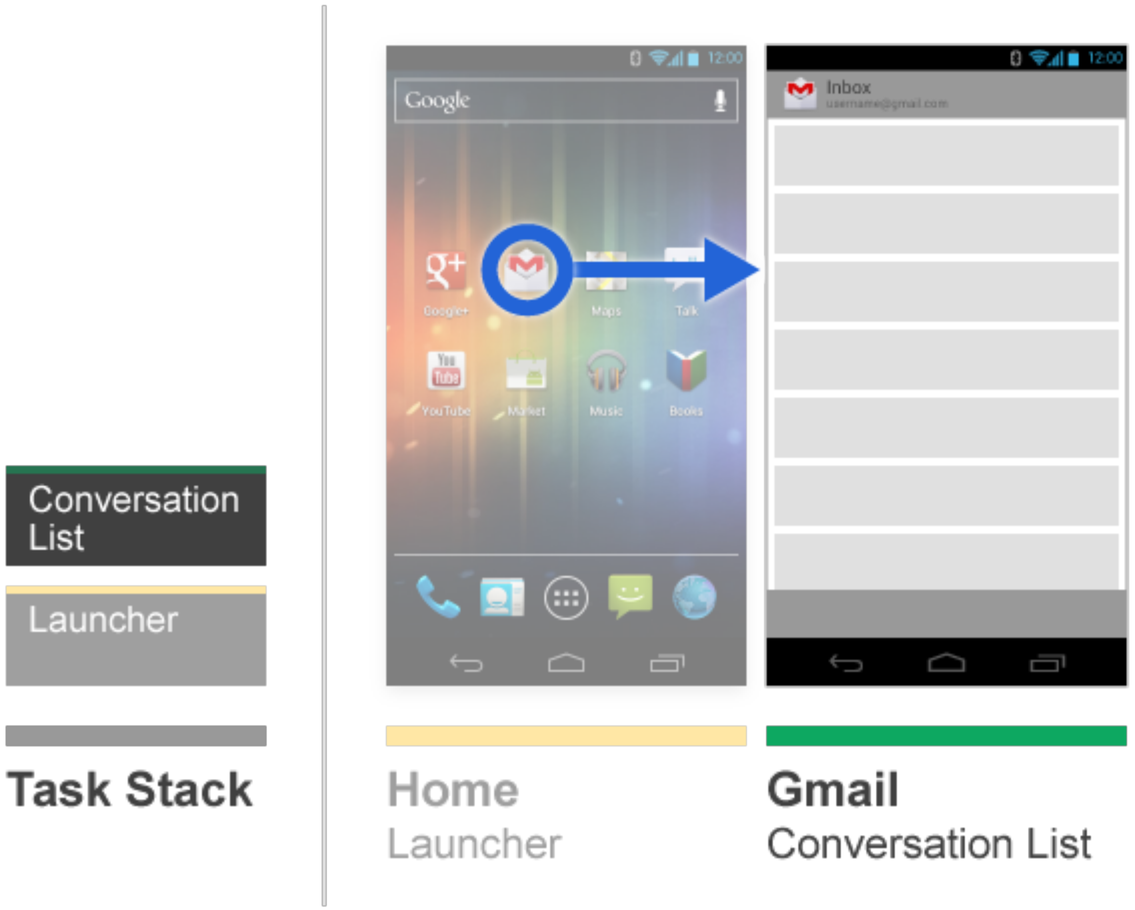
Task Stack



Home
Launcher

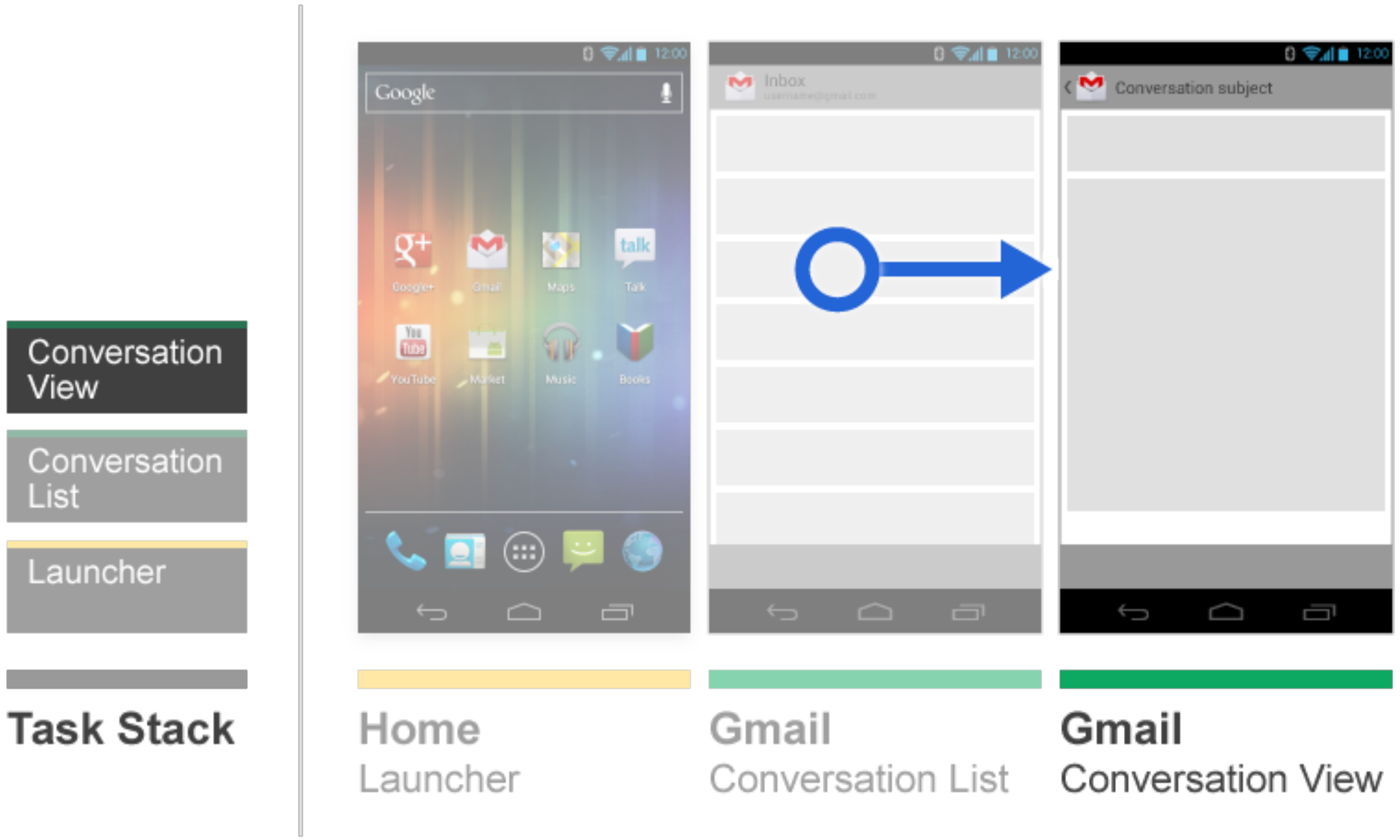
The elephant in the room: Notifications

A common use case: follow notification directly from one app to another



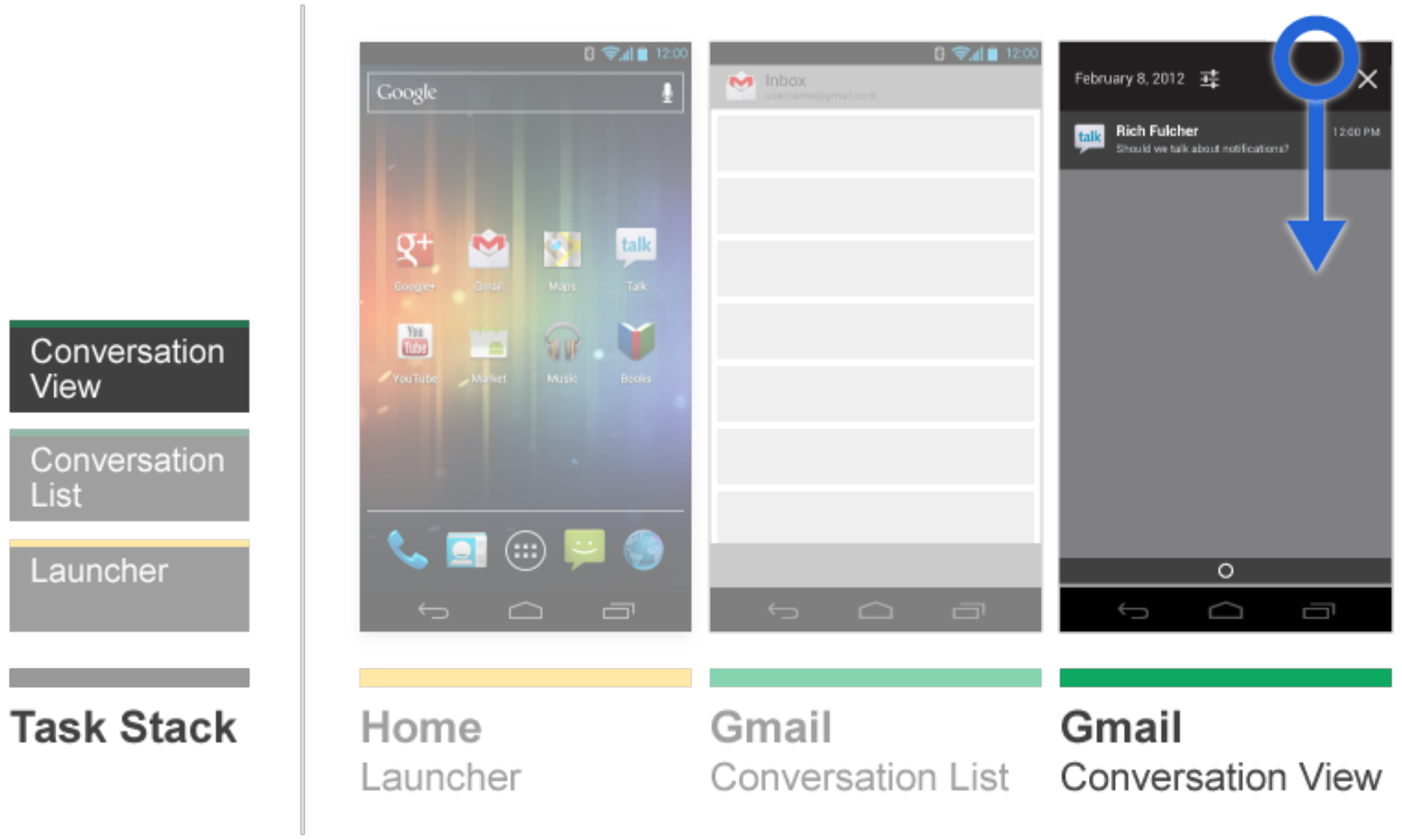
The elephant in the room: Notifications

A common use case: follow notification directly from one app to another



The elephant in the room: Notifications

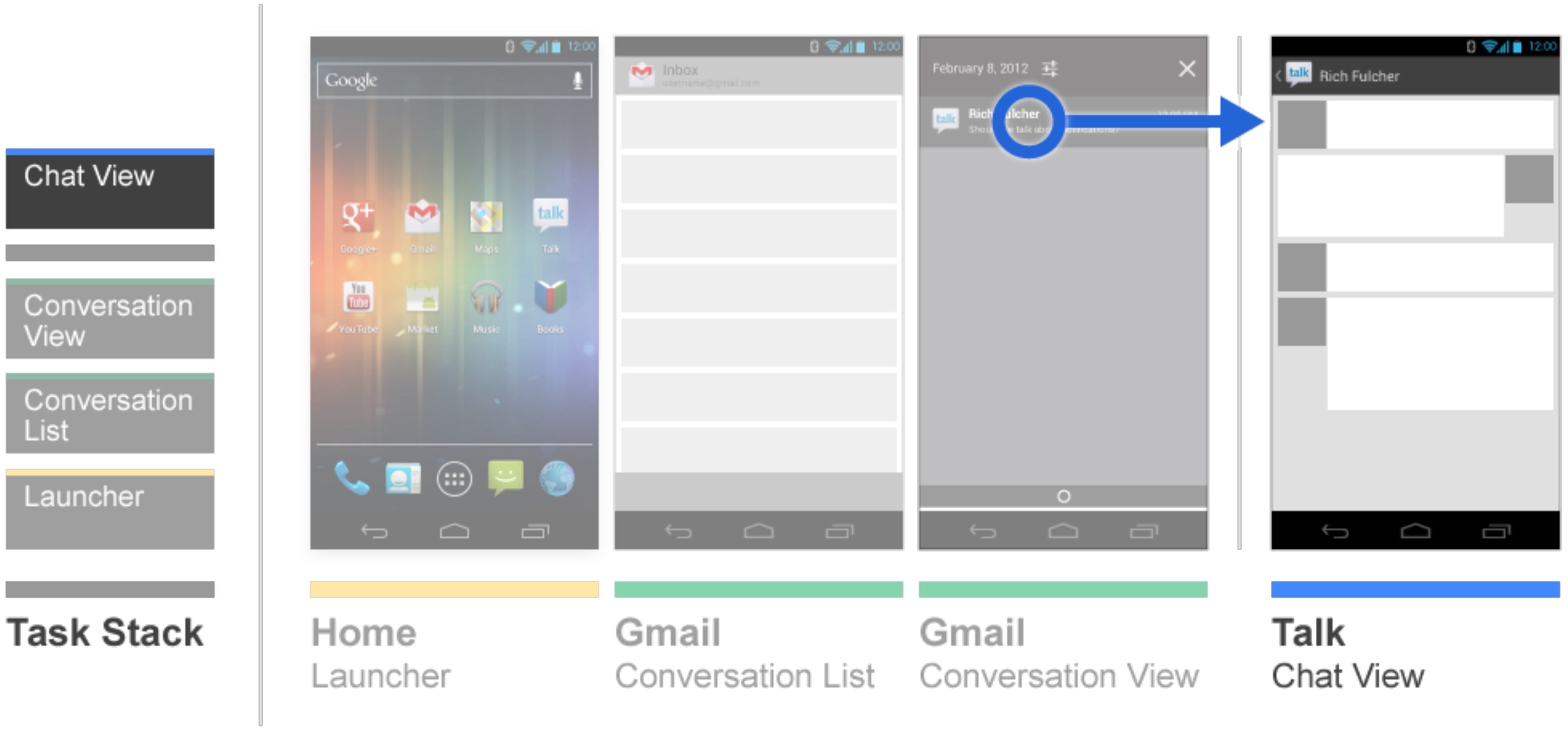
A common use case: follow notification directly from one app to another



Rich Fulcher 12:00 PM
Should we talk about notifications?

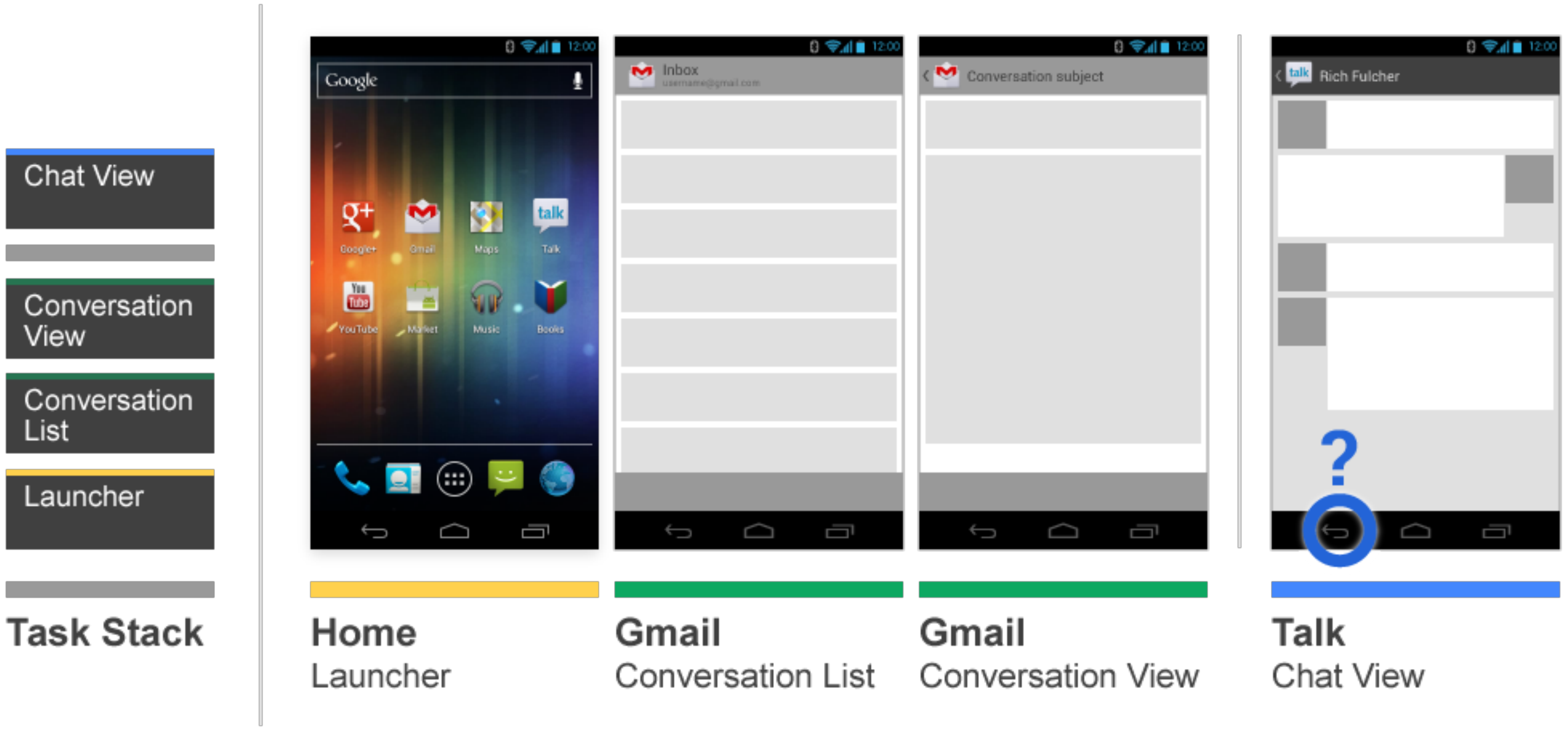
The elephant in the room: Notifications

A common use case: follow notification directly from one app to another



The elephant in the room: Notifications

A common use case: follow notification directly from one app to another



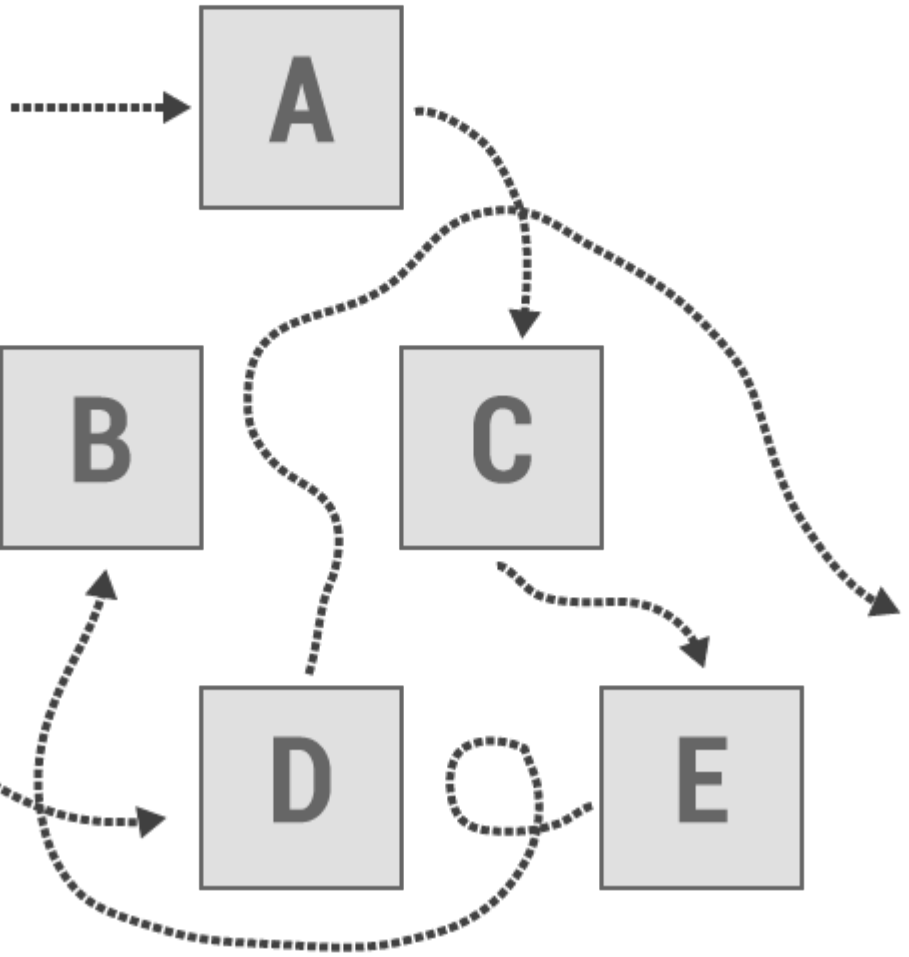
The elephant in the room: Notifications

But knowing what you now know,
we can't just hop back to the previous task.

"Why did THAT happen?"

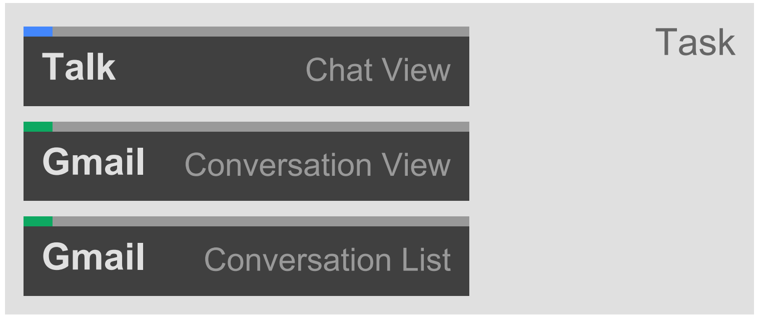
Weird task state often doesn't manifest until you come back to the task later

- Hours?
- Days?

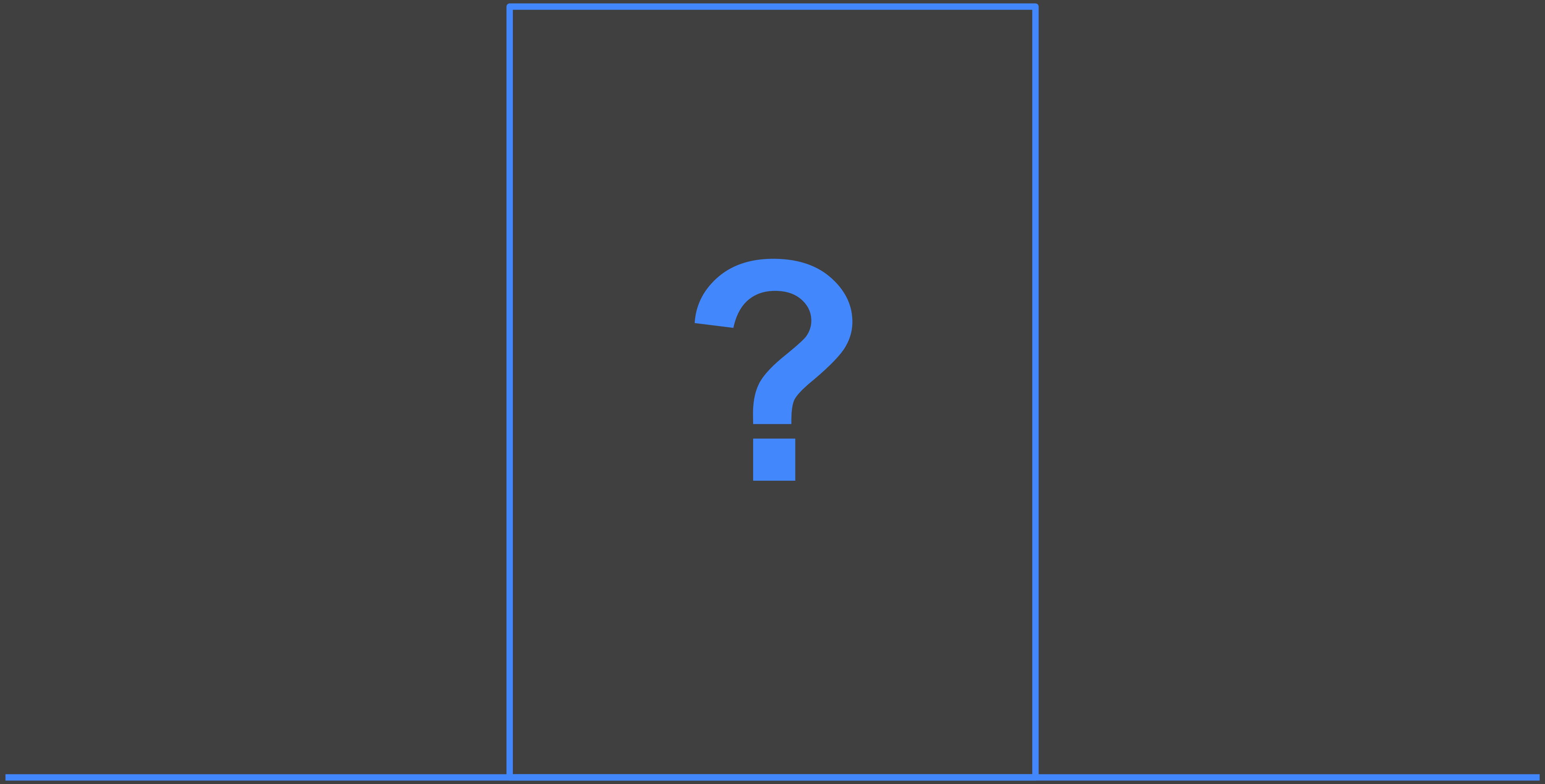


What task do Notifications show up in?

We've already said that Back should act on the **current** task
...so do notifications show up on the current task?



That doesn't make any sense.



“Entering or exiting through a doorway serves as an 'event boundary' in the mind, which separates episodes of activity and files them away.”

"Recalling the decision or activity that was made in a different room is difficult because it has been compartmentalized."

Gabriel Radvansky
University of Notre Dame Department of Psychology

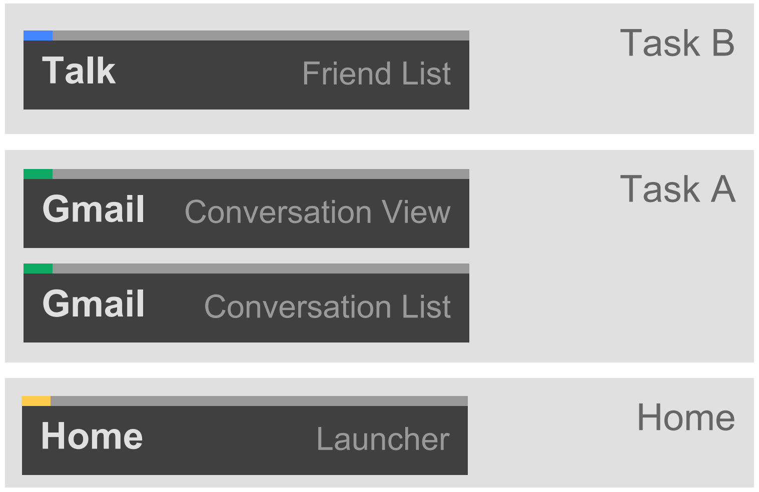
What task do Notifications show up in?

Can we cheat?

Start in a new task, but don't set `FLAG_ACTIVITY_TASK_ON_HOME`!

But we still have some requirements:

- We still need to avoid the "open in new tabs" problem.
- We don't want to leave the app that posted the notification in a bogus state
- What about the rest of the existing task?



What task do Notifications show up in?

Maybe a dirty trick?

- Set `android:taskAffinity="" android:excludeFromRecents="true"` on the target of the notification

- No affinity? No problem!
  - ...not quite.

- What if you get a phone call or switch tasks?
  - Hope you saved a draft...
  - Where does Recents take you?

(This approach is actually useful in some cases after all)

The solution?

Navigating into an app from a notification replaces the target task entirely.

- Conceptually notifications become a shortcut for:
    - Press Home
    - Press Recents and swipe away the current task for the target app
    - Open the app
    - Find the shortest path to what the notification was trying to tell you

- TaskStackBuilder gives an easy way to accomplish this

The implications

Yes, this means you can't use Back from answering a notification to switch to the previous task anymore.

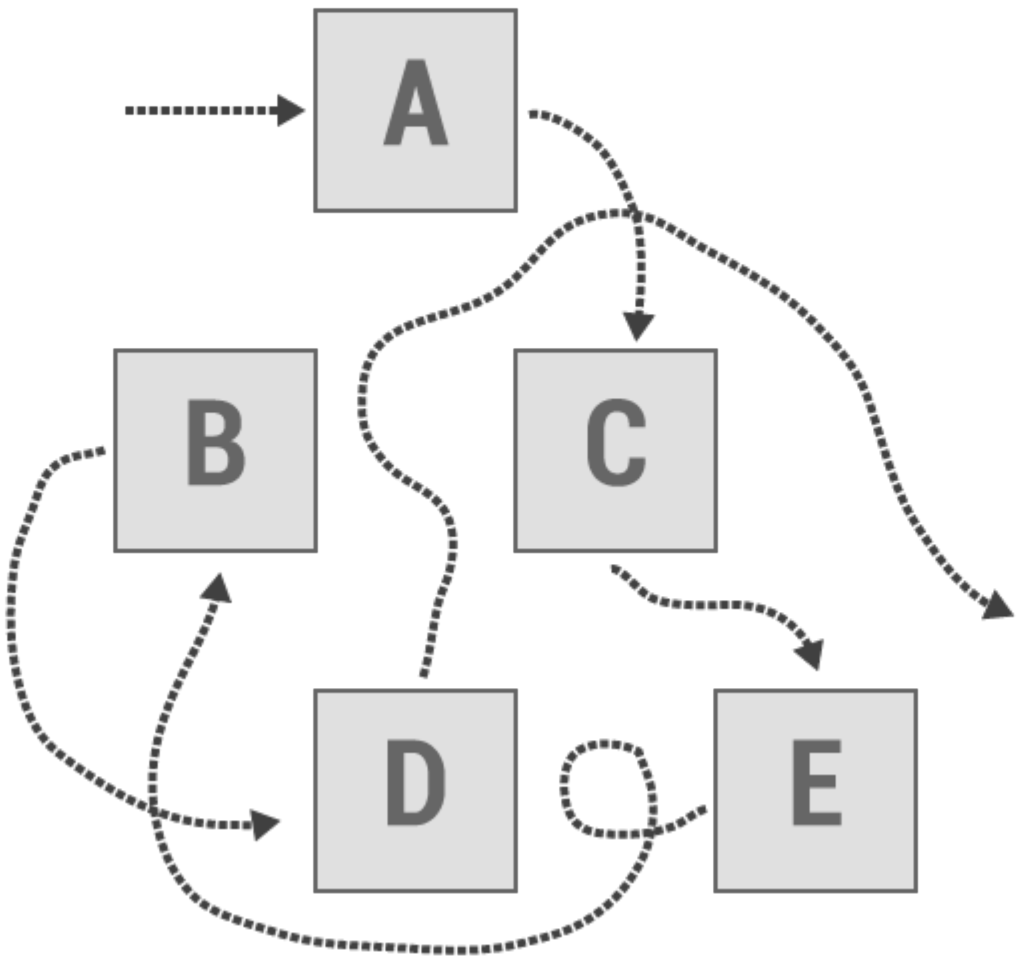
- Recents to the rescue!
  - Most recent task is always sorted closest to the Recents button
  - Reinforces a consistent idea of tasks and cross-task navigation

- Telling the story with animation in Jellybean

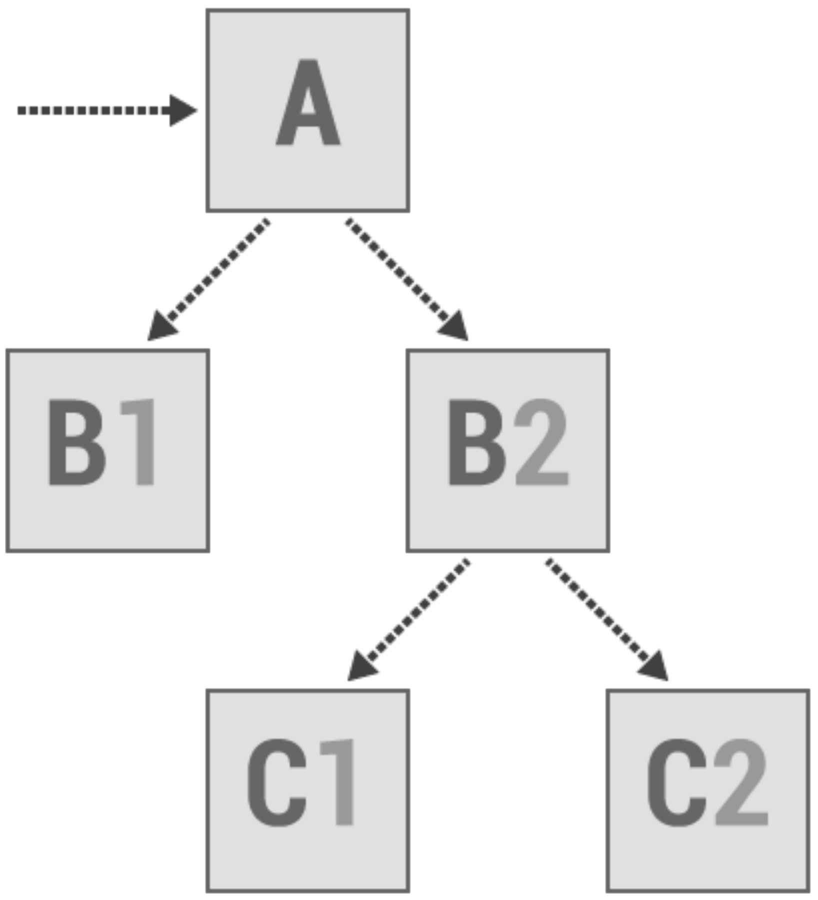
Things to Remember

3 takeaways

Think about the structure of your app early on



vs



Don't test your app navigation in isolation

- Test all navigation in and out of your app:
  - Notifications
  - Widgets
  - Intent fulfillment (e.g. Share)

- Test over a number of days
  - Let tasks accumulate and deepen

Users *always* win when apps behave consistently

- No need for users to learn how to navigate your app
  - Users leverage prior knowledge
  - Don't need to spend time teaching users
- Give users a sense of mastery
  - Immediately positive associations with your app
  - Able to concentrate energy on core of your app
- Device feels more like one cohesive experience
  - Lifts satisfaction/confidence across the board

Thank You!

Richard Fulcher, Android Interaction Designer
Adam Powell, Android Framework Engineer

Google Developers