Vitali Levit
Senior Product Designer (UI× UX × Design Systems)

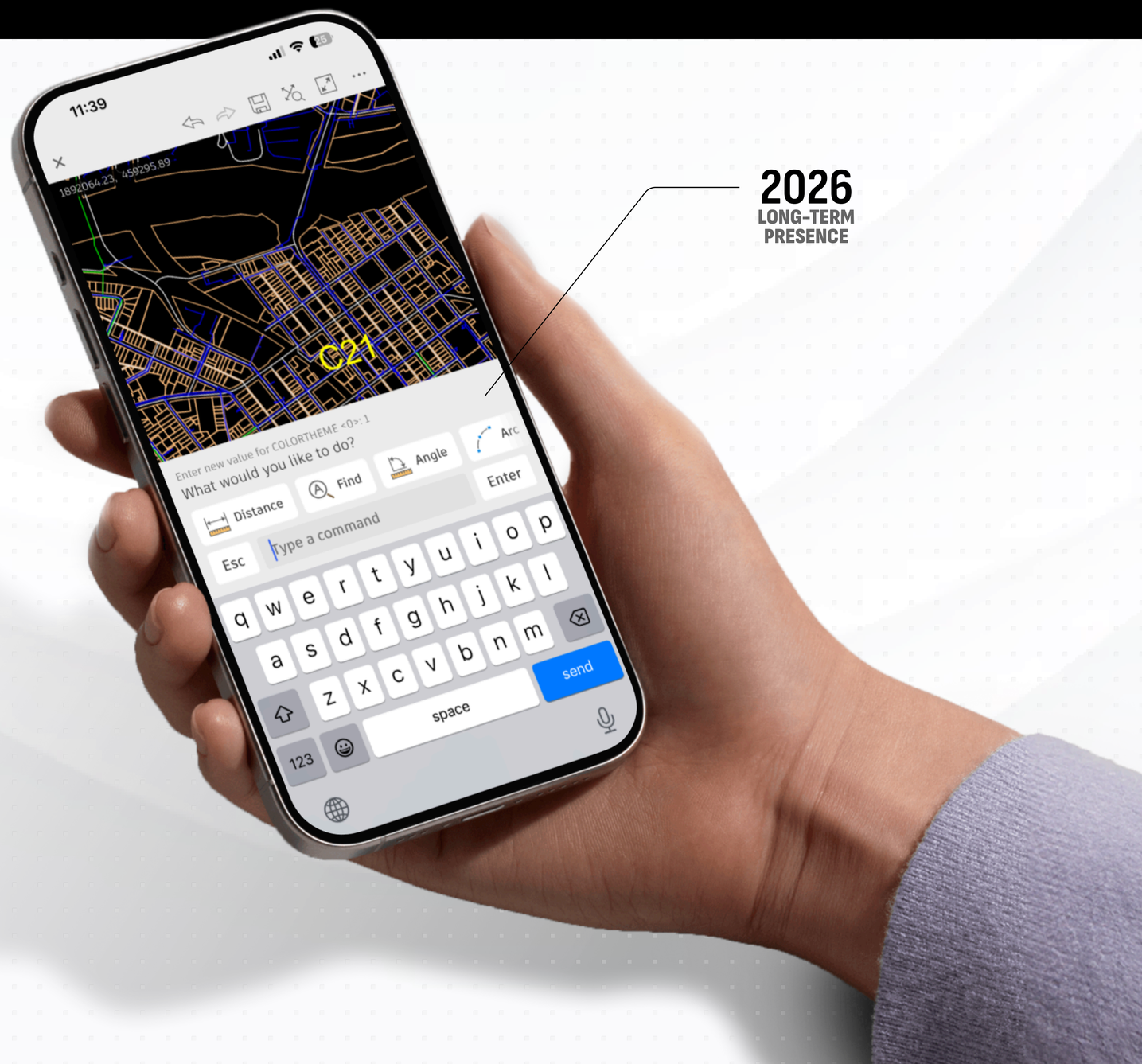
AUTODESK

How I reintroduced a familiar expert workflow into AutoCAD Mobile and turned a risky idea into durable product value.

Reintroducing Familiar Workflows

For AutoCAD Desktop users, command line is a very familiar and powerful workflow. But on mobile, it sounded risky at first - mobile products are usually expected to be simpler, more visual, and less command-driven.

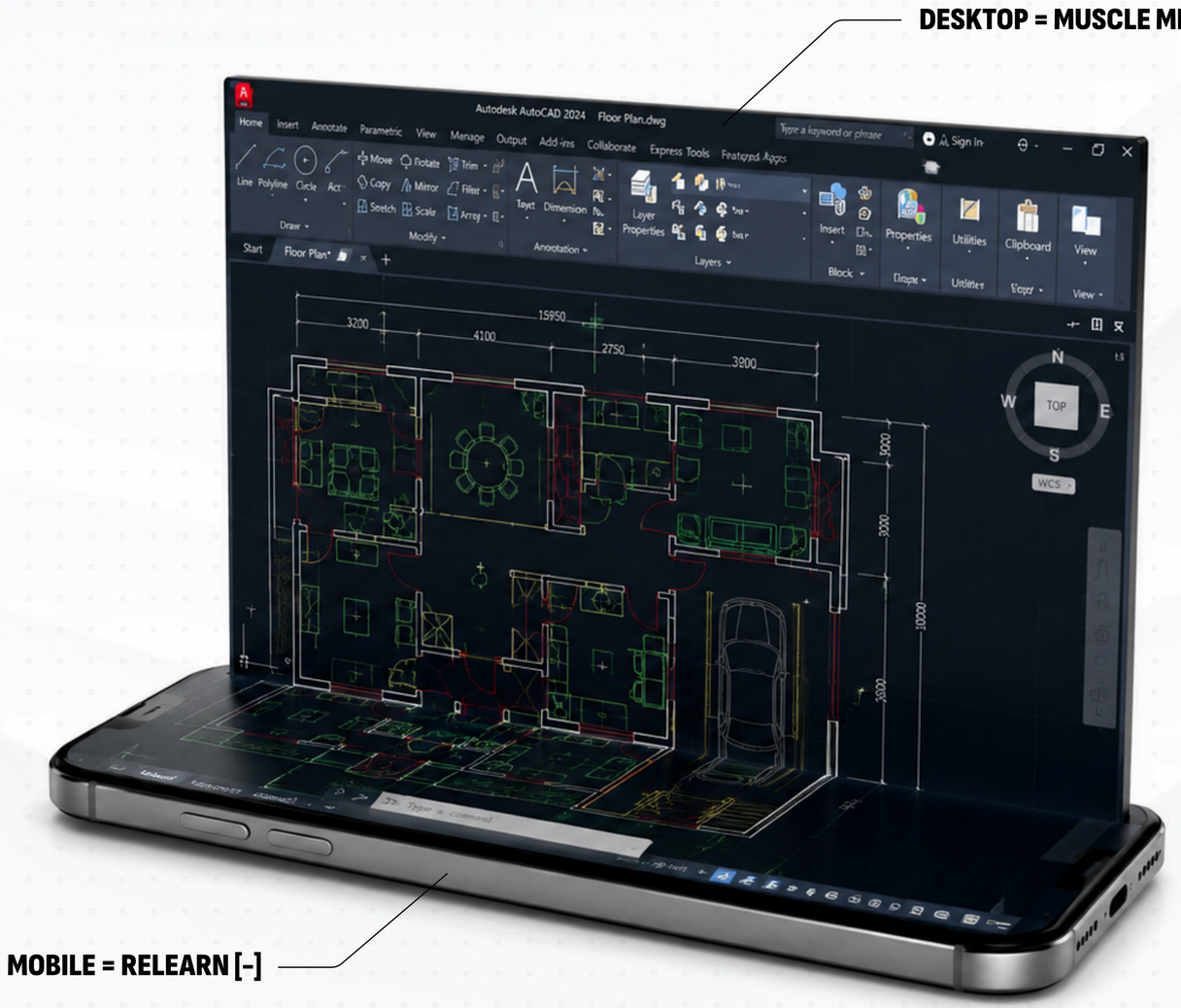
What makes this case interesting is that the feature is still visible in the App Store listing more than four years later = It solved a real workflow problem.



Broken Workflows, Not Missing Features

Experienced AutoCAD users already had years of habits, muscle memory, and expectations from Desktop. When they moved to mobile, the product had functionality, but the interaction model felt different enough that they had to relearn parts of the workflow.

Users didn't want a different AutoCAD on mobile.
They wanted AutoCAD.



Research Reframed the Problem

This came from user research.

I participated in multiple sessions with professional AutoCAD users, and in some cases helped organize and observe those sessions.

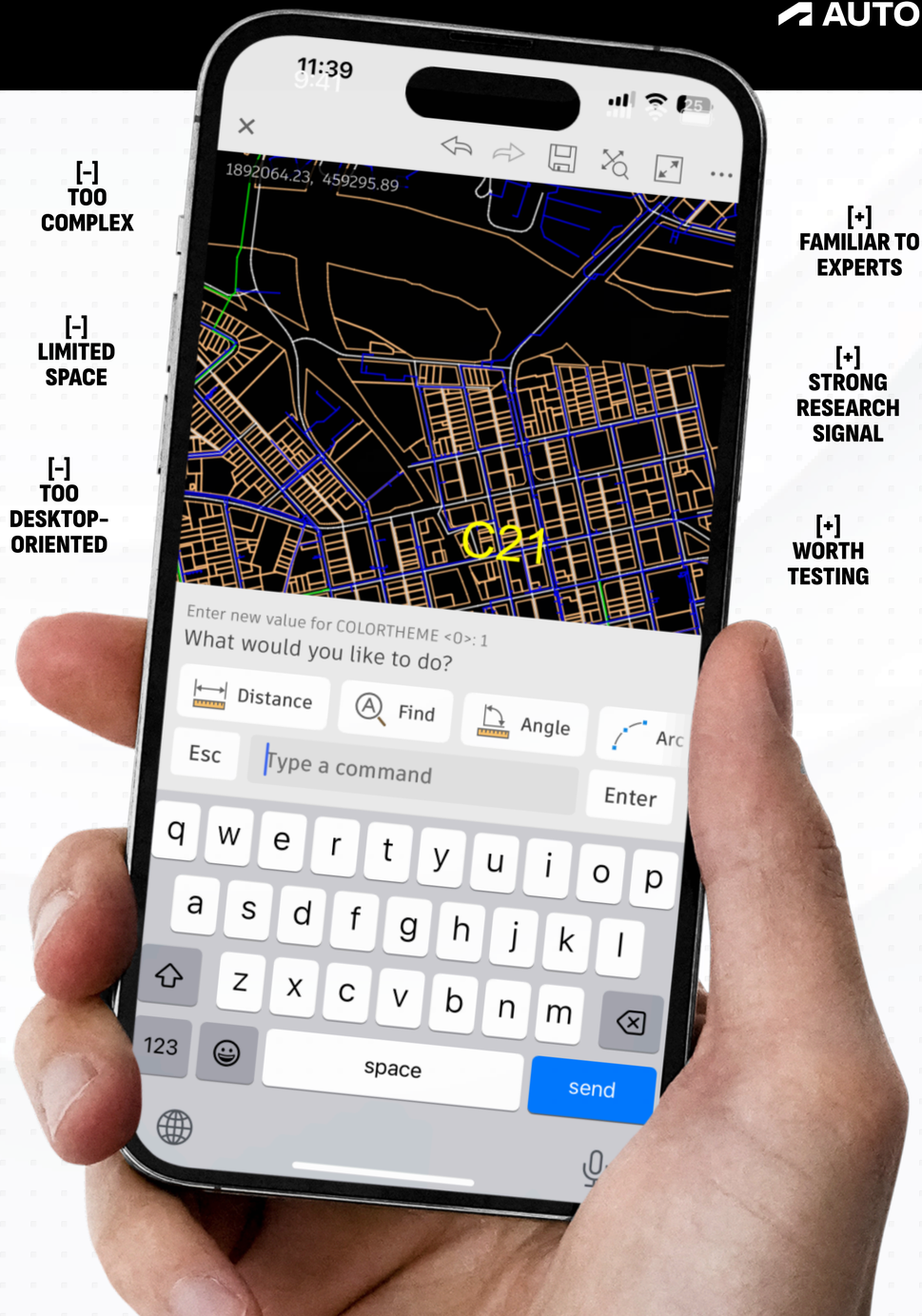
What was important for me was not only what users said, but how they behaved: where they slowed down, where they hesitated, and where mobile interaction patterns did not match their existing mental model.

AUTODESK

We Challenged a Desktop-Only Assumption

Command line felt like a desktop pattern. There were reasonable concerns that it could be too complex for mobile, take too much space, or confuse less experienced users.

So together with two engineers, I explored it as an initiative. We were not just executing a roadmap item. We were testing whether a risky desktop assumption could become a useful mobile workflow.



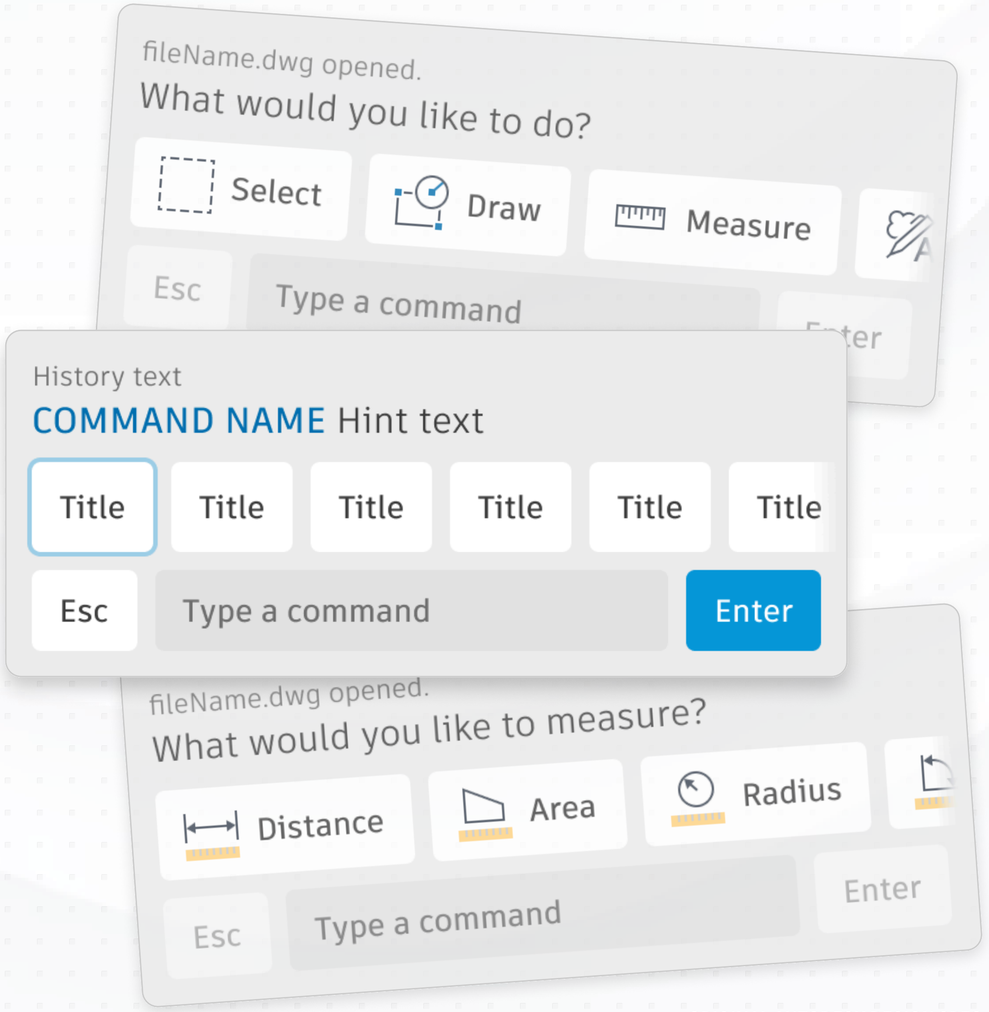
Designed as a System, Not a One-Off Feature

When we moved into the solution, we did not just copy the desktop command line.

We had to rethink it as a mobile system: different states, widgets, layout behavior, scaling rules, platform constraints, and interaction patterns.

For me personally, this was one of the early moments where I started moving from designing screens to designing systems.

This way of thinking later became very important in my work on design systems and scalable UI architecture.



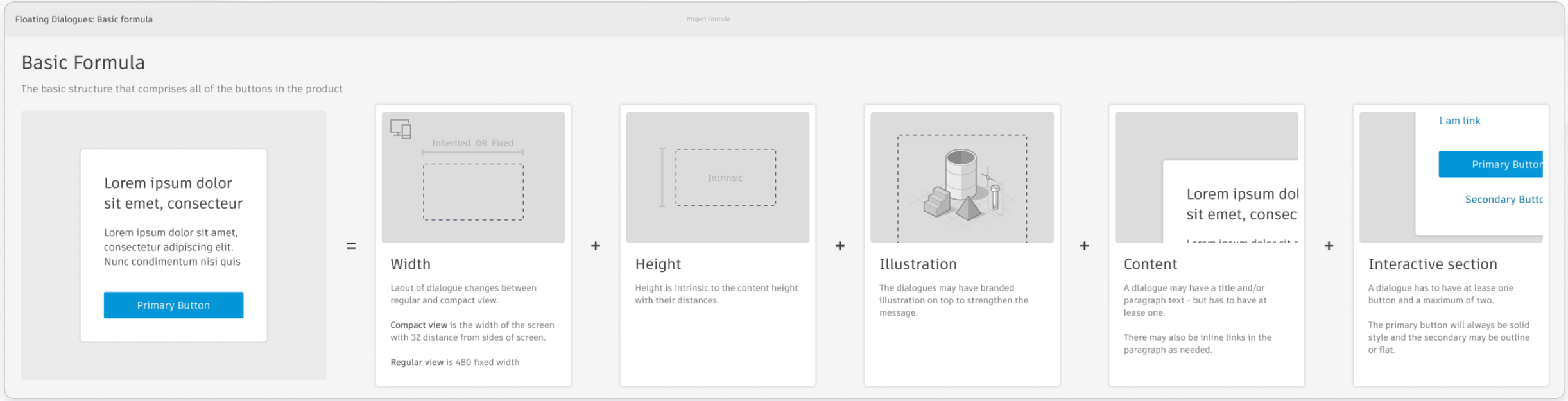
ORIGINAL AUTODESK DESIGN SPEC

The Feature Shipped – and Still Shapes My Thinking

The feature shipped, stayed in the product, and remains part of AutoCAD Mobile more than four years later.

For me, that longevity matters. It means the decision was not just visually relevant at the time – it solved a durable product problem.

This project became a foundation for how I think about product design today: understanding real user workflows, making careful product decisions, working closely with engineering, and turning UI into systems rather than isolated screens.



ORIGINAL AUTODESK DESIGN SPEC

Appendix

Extra materials: selected screenshots from the original AutoCAD Mobile design spec.
Disclaimer: this is proof of the spec, not the full document.

Neutral state

Component:
Neutral state;
Category opened state;
Opened keyboard state;

No any active command, nothing is selected on the canvas

- command line access: app
- hint/prompt: app
- prompt history (fresh opened file): app
- prompt history (fresh opened file): fabric
- shortcuts: app
- shortcuts categories: app
- enter button: app
- esc button: app

Selected state

Component:
Category opened state;
Opened keyboard state;

One or multiple geoms are selected on the canvas

1. command line access: app
2. hint/prompt: fabric
3. prompt history: fabric
4. shortcuts: app
5. enter button: app
6. esc button: app

Command active state

Component:
Active state;

A command is active

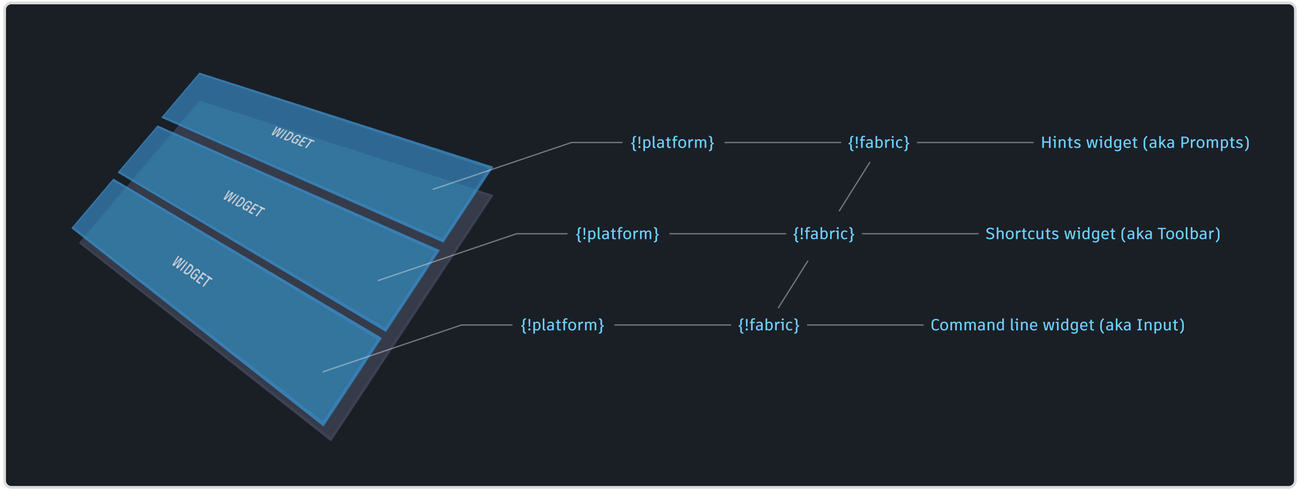
1. command line access: app + fabric
2. hint/prompt: fabric
3. prompt history: fabric
4. comamnd name: app + fabric
5*. keywords (if avaialble): fabric
6. enter button: fabric
7. esc button: fabric

Widgets

This is a modular component build out of widgets

Modular: The component represents a collection of widgets designed to work together. The widgets may be enabled or disabled, also replaced with different widgets. The component should be able to adjust for different scenarios such as: devices types, orientations, canvas modes. See "Screen classes" for iOS and "Breakpoints" for Android.

Editor mode



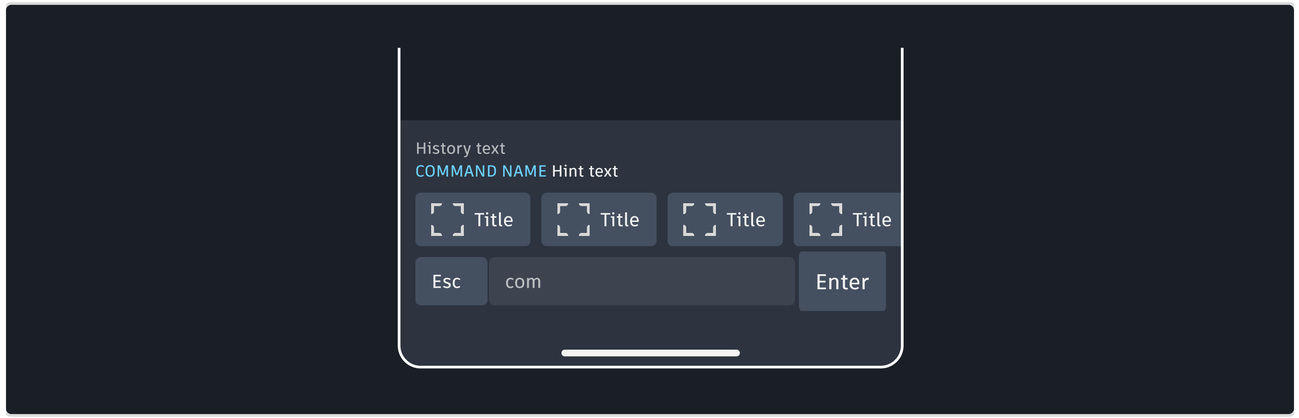
1. Example: Editor mode:
   3 widgets presented.
2. "Hints widget" comes with 2 sub-widgets:
   History bar and System prompts.
3. "Shortcuts widget"
   displays quick tools access buttons (aka the Toolbar).
4. "Command line" widget
   comes with an input field to send system commands together with "esc" and "enter" buttons.

Viewer mode

Styles

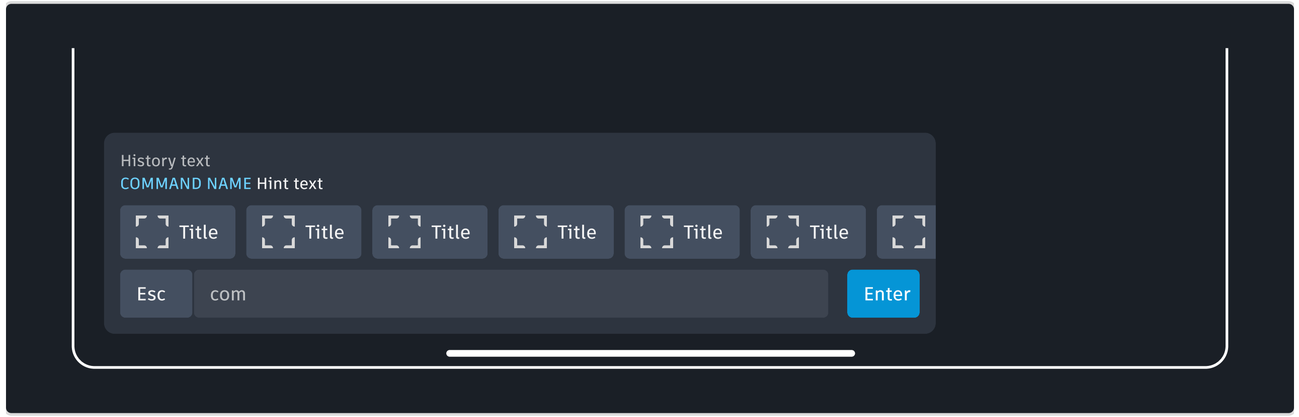
We use different styles for different cases

Docked



1 Docked style:
Seating tight inside a screen. Goes edge to edge.
Preferred for: Portrait phones. See “Screen classes” and “Breakpoints” to learn more.

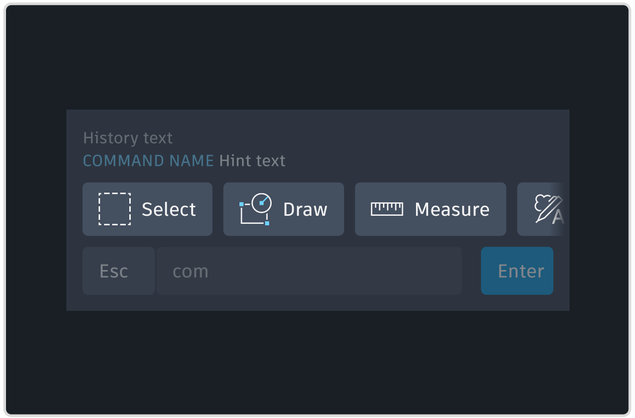
Floating



Shortcuts widget: States

Shortcuts widget VA states

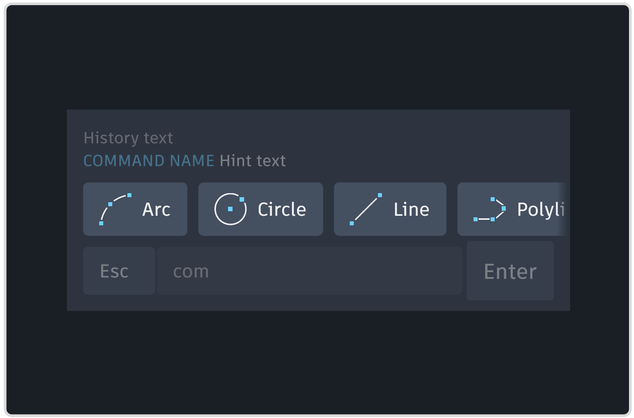
Neutral state



1 {!platform}
Display “shortcuts categories” and commands.

2 {!fabric}{!platform}
Any button press will open relevant category of commands or activate a command depends on a button purpose.

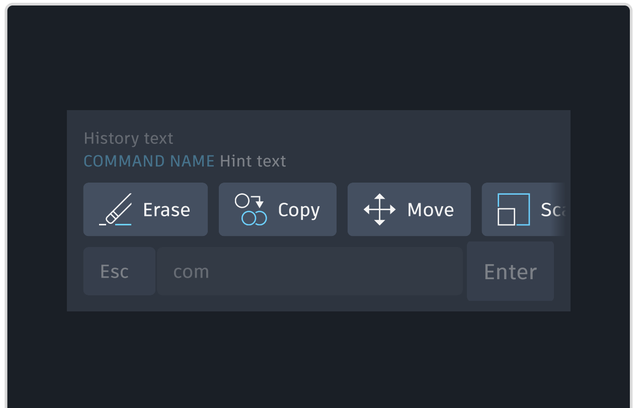
Neutral state / Shortcuts category opened



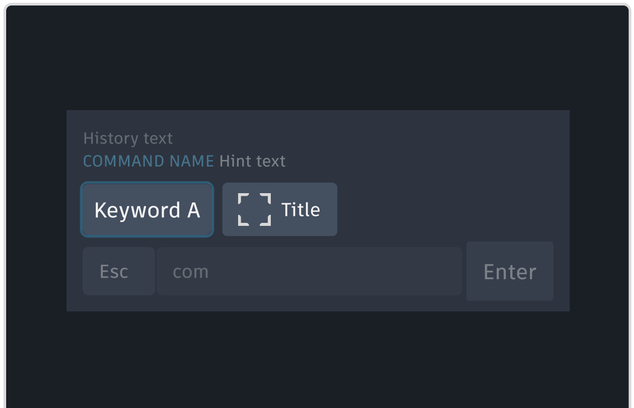
1 {!platform}
Display “shortcuts” of a selected category.

2 {!fabric}
Any button press will activate relevant command.

Selected state



Command active state

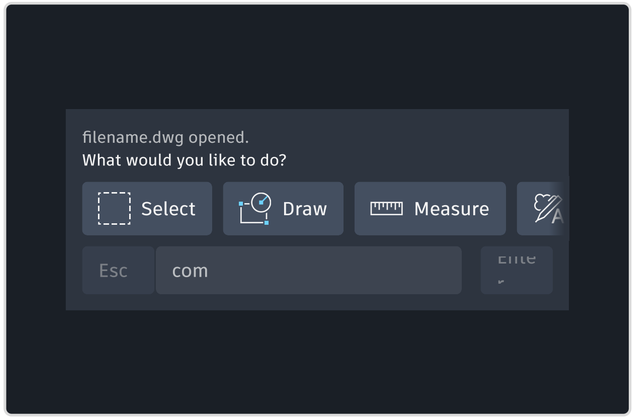


Editor vs Viewer

Layout changes between "Editor" and "Viewer" modes

⚠️ These are high-level examples. For more layout changes see "Screen classes" for iOS and "Breakpoints" for Android.

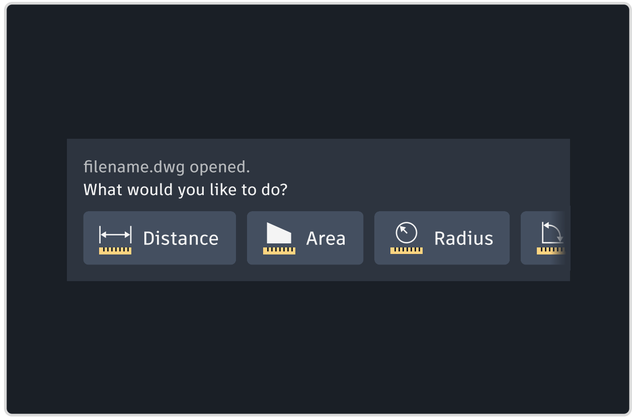
Editor: Neutral state



1 Default.

2 “Editor Light”:
In some cases this view is disabled. See “Screen classes” for iOS and “Breakpoints” for Android.

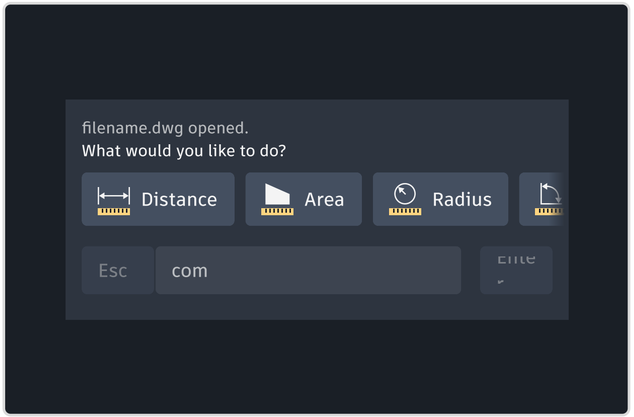
Viewer: Neutral state



1 Layout change:
No “Command line widget”.

2 (!) Based on the “Viewer” mode definition:
Tools-set change.

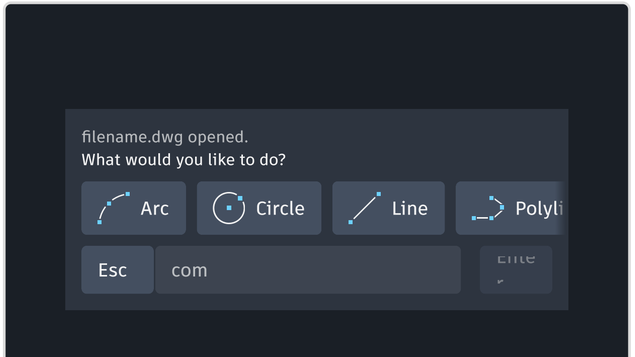
ViewerForEditor: Neutral state



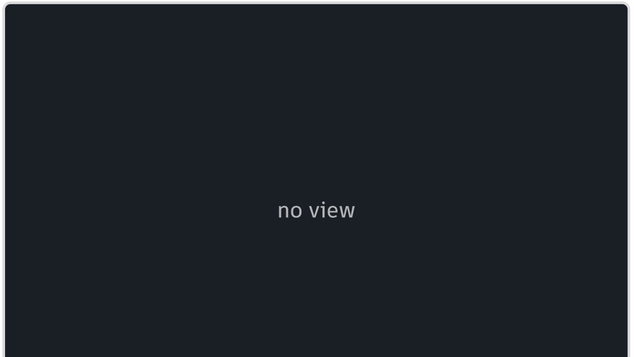
1 (!) Based on the “ViewerForEditor” mode definition:
Same as “Viewer”.

2 (!) Based on the “ViewerForEditor” mode definition:
Tools-set change.

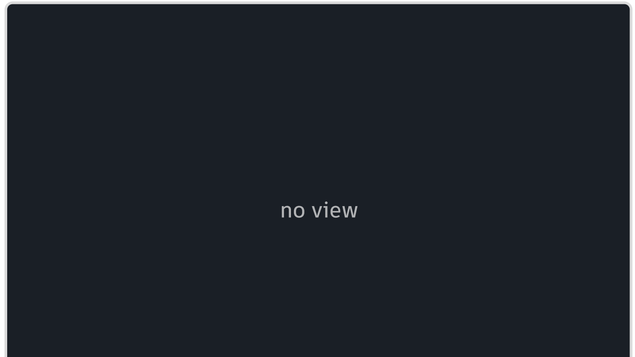
Editor: Shortcuts category opened



Viewer: Shortcuts category opened



ViewerForEditor: Shortcuts category opened



Initial strings for localization.

All custom strings: History bar, Prompts bar, Command line input.

Feature Owner review.

@Vitali Levit, AutoCAD Mobile XD.

Platforms involved.

Same for all the platforms. What differen: “Screen classes” for iOS and “Breakpoints” for Android.

Unique OS behaviors.

Android soft-key “Back” defined separately.

Devices: Phones, Tablets.

Support both phones and tablets by changing the layouts. See “Screen classes” for iOS and “Breakpoints” for Android.

Cross platform implications: web, desktop, mac, android, UWP.

iOS and Android, Chrome OS.

Offline / Online.

Should work the same as online as offline.

Free / Bundle / Students / Pro / Pro plus user / Trial.

Depends on the “Canvas mode”. Supported modes: Editor, ViewerForEditor, Viewer.
Support all by changing the layouts. See “Screen classes” for iOS and “Breakpoints” for Android.

File types: DWG / DXF / PDF / PNG / JPEG / JIF / BMP / TIF.

Depends on the “Canvas mode”. Supported modes: Editor, ViewerForEditor, Viewer.

2D

Display for 2D canvas.