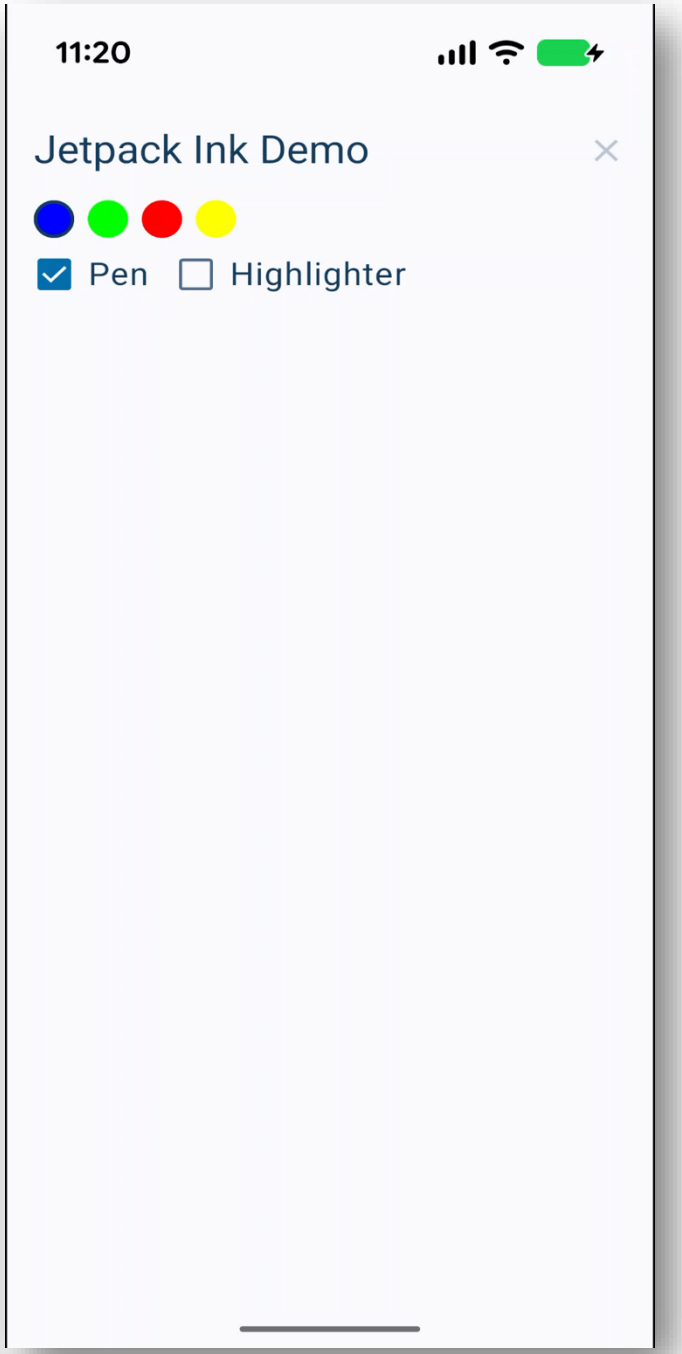
Supporting stylus input on Android using Jetpack Ink API

Thomas Künneth

Experts

Photo Art by Monika Künneth



https://github.com/tkuenneth/JetpackInkDemo

Experts

SnappMobile

Do styluses matter?

- Very popular among Samsung users
- Standard feature on modern ChromeOS devices, especially in education and creative markets
- Elsewhere, stylus support is sporadic and inconsistent

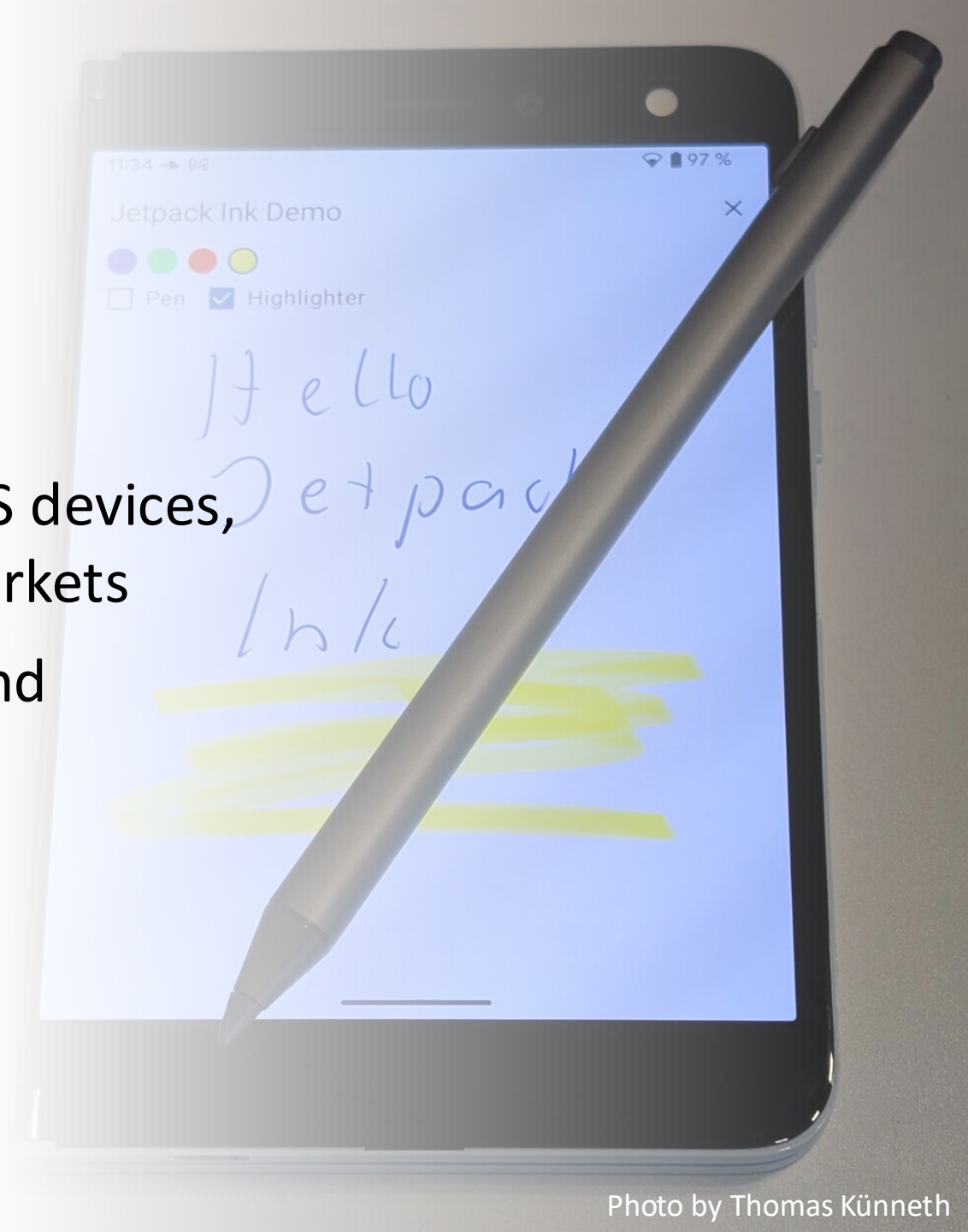
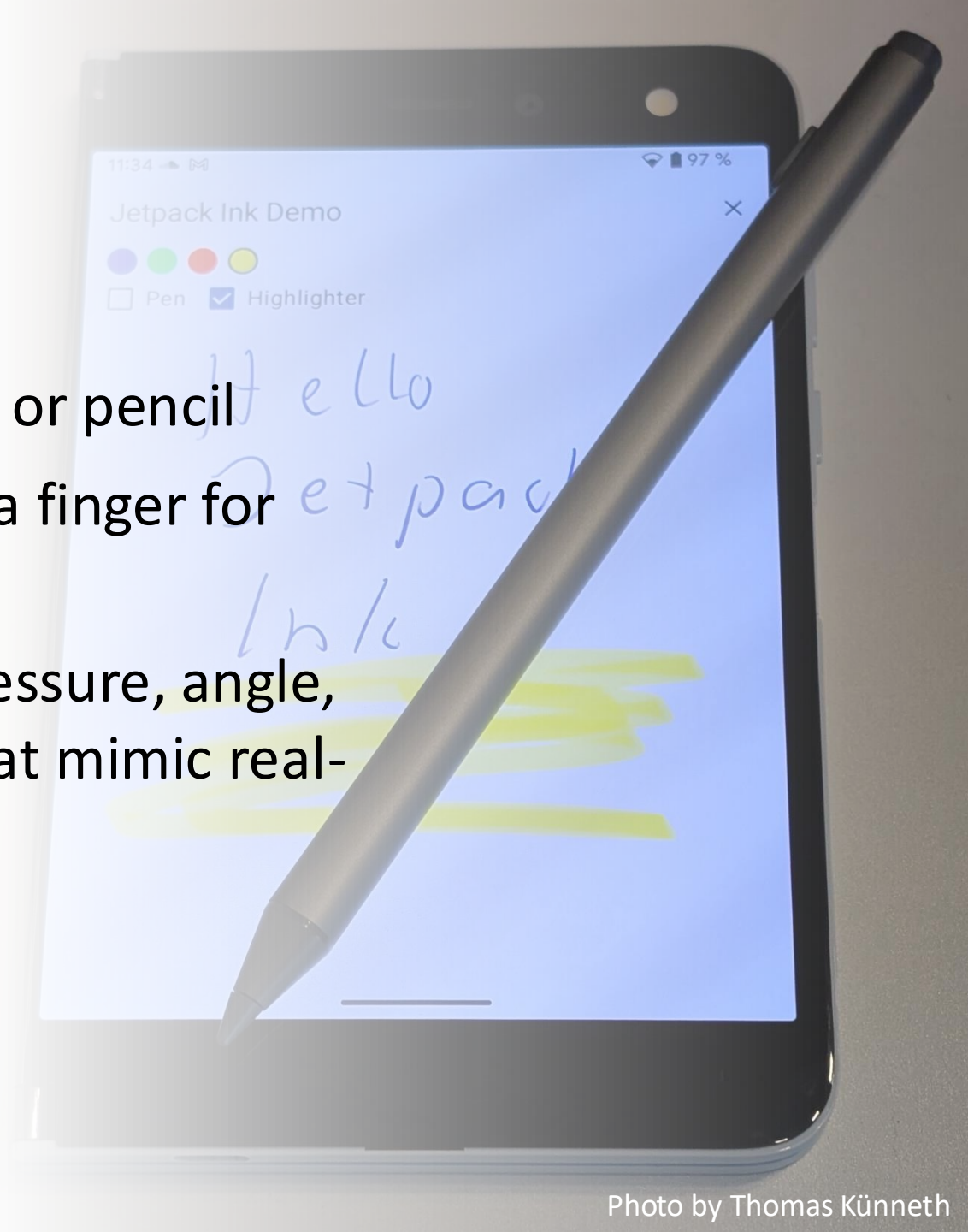


Photo by Thomas Künneth

Experts

A premium experience

- Using a stylus feels like using a real pen or pencil
- A stylus offers much finer control than a finger for detailed tasks
- Styluses provide rich data, including pressure, angle, and tilt, allowing for realistic strokes that mimic real-world tools

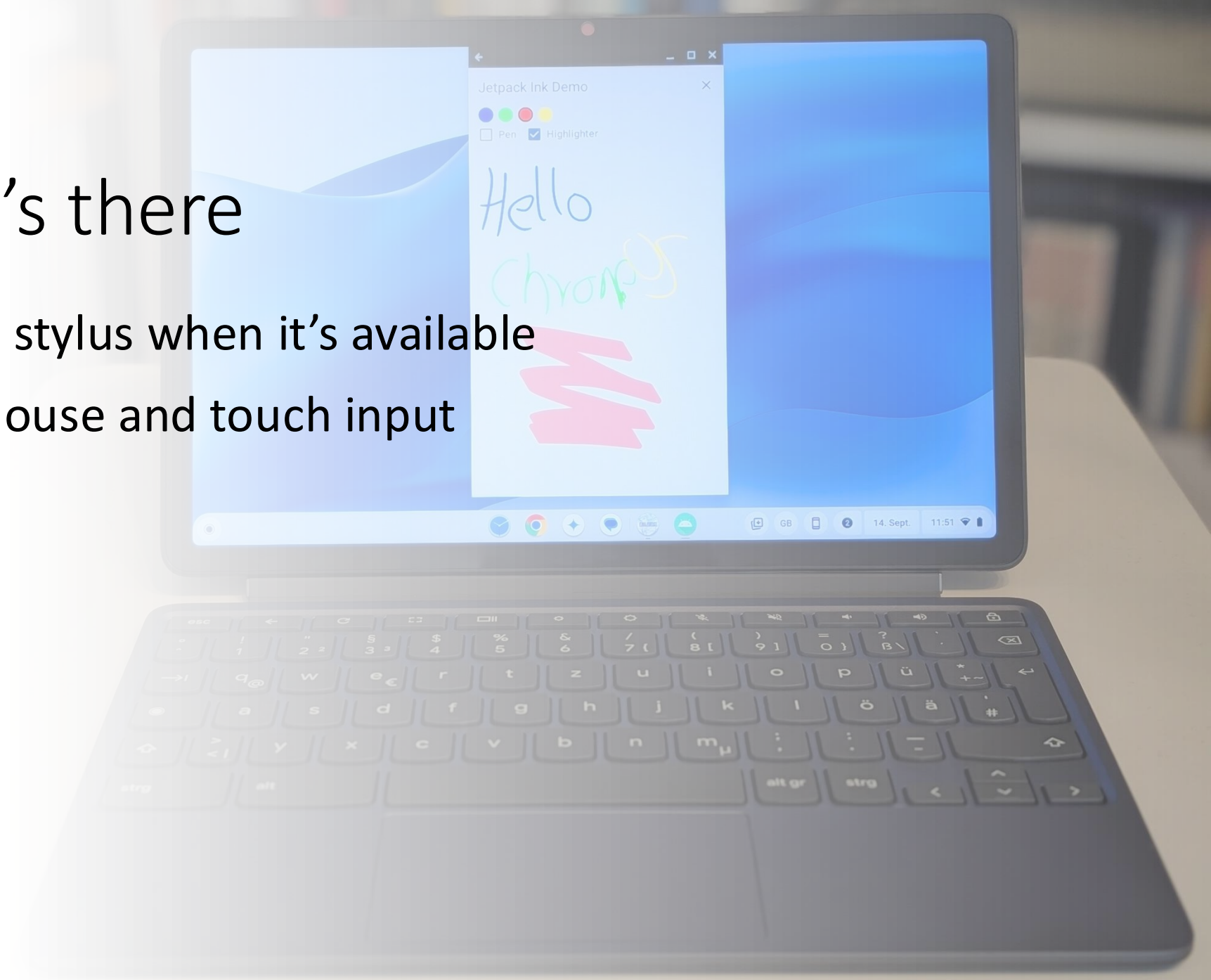


Experts

Photo by Thomas Künneth

Support what’s there

- Take advantage of a stylus when it’s available
- But don’t neglect mouse and touch input



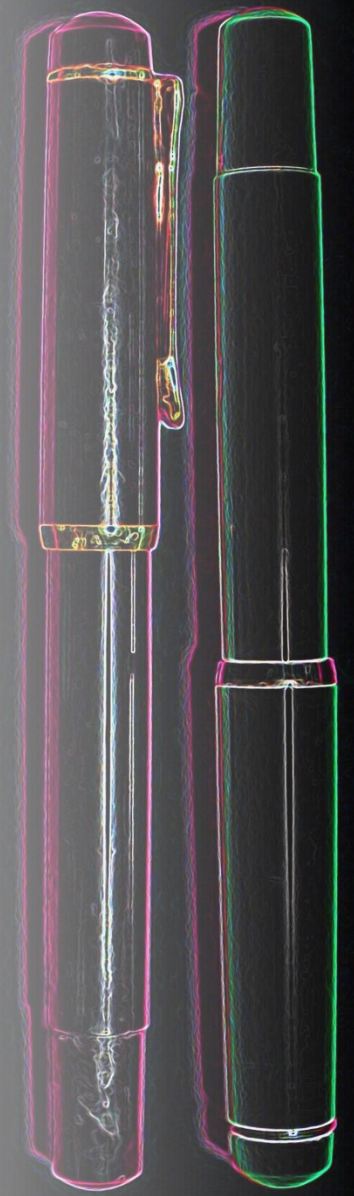
Experts

Photo by Thomas Künneth

Jetpack Ink API

- Abstracts away the complexities of graphics and geometry, allowing you to focus on your app's unique inking features
- Built-in low latency support and optimized rendering ensure a smooth and responsive inking experience

- First release 1.0.0-alpha01 was made available early October 2024
- As of September 2025, current release is 1.0.0-alpha06
- Google currently updating docs and samples



Experts

Photo Art by Monika Künneth

- Leverages benefits of styluses
- Also works with finger and mouse input
- Modular design allows you to pick and choose the components you need

- **Strokes**: Foundation of the library; represents ink input and its visual representation
- **Brush**: Defines the look of your strokes: color, thickness, and style
- **Rendering**: Efficiently draws the ink, ensuring it looks great in both Jetpack Compose and Android Views

- **Geometry**: Provides the building blocks for creating tools like erasers and selection areas
- **Authoring**: Captures touch input and turns it into smooth, responsive ink on the screen with lowest possible latency
- **Storage**: Helps saving and loading strokes

```
dependencies {
    implementation(libs.androidx.ink.authoring)
    implementation(libs.androidx.ink.authoring.compose)
    implementation(libs.androidx.ink.brush)
    implementation(libs.androidx.ink.geometry)
    implementation(libs.androidx.ink.nativeloader)
    implementation(libs.androidx.ink.rendering)
    implementation(libs.androidx.ink.strokes)
    implementation(libs.androidx.ink.storage)
    implementation(libs.androidx.input.motionprediction)
}
```

Native C++ libraries that are essential for the core functionality of the Ink API, including rendering and geometry calculations

Advanced algorithms to predict and stabilize user input, significantly reducing latency and improving the accuracy of digital ink strokes for a smoother drawing experience

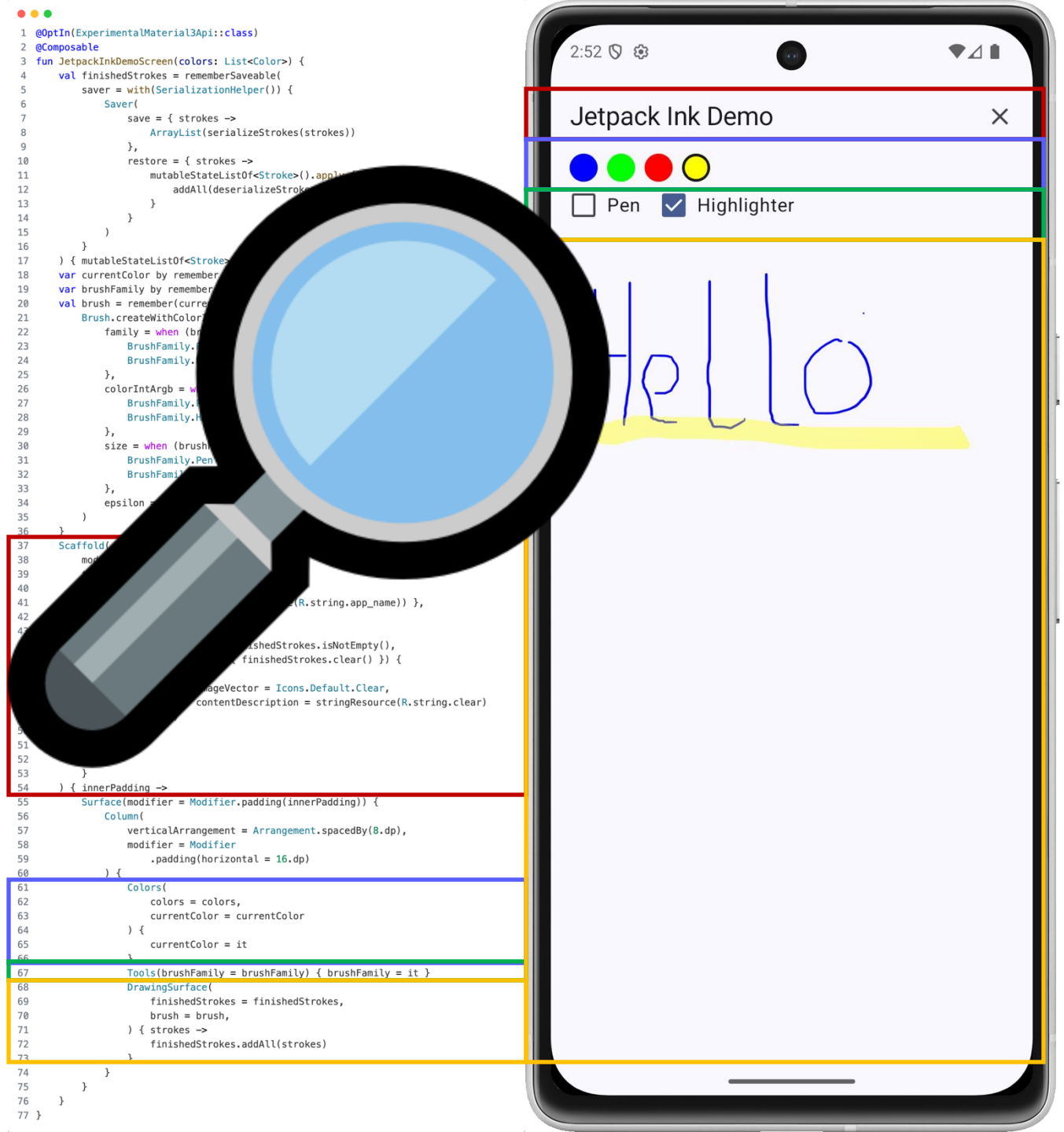
```
[versions]
inkVersion = "1.0.0-alpha06"
inputMotionprediction = "1.0.0-beta06"

[libraries]
androidx-ink-authoring = { module = "androidx.ink:ink-authoring", version.ref = "inkVersion" }
androidx-ink-authoring-compose = { module = "androidx.ink:ink-authoring-compose", version.ref = "inkVersion" }
androidx-ink-brush = { module = "androidx.ink:ink-brush", version.ref = "inkVersion" }
androidx-ink-geometry = { module = "androidx.ink:ink-geometry", version.ref = "inkVersion" }
androidx-ink-nativeloader = { module = "androidx.ink:ink-nativeloader", version.ref = "inkVersion" }
androidx-ink-rendering = { module = "androidx.ink:ink-rendering", version.ref = "inkVersion" }
androidx-ink-strokes = { group = "androidx.ink", name = "ink-strokes", version.ref = "inkVersion" }
androidx-ink-storage = { group = "androidx.ink", name = "ink-storage", version.ref = "inkVersion" }
androidx-input-motionprediction = { group = "androidx.input", name = "input-motionprediction", version.ref = "inputMotionprediction" }
```

```kotlin
class MainActivity : ComponentActivity() {

    @OptIn(ExperimentalMaterial3Api::class)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            MaterialTheme(
                colorScheme = defaultColorScheme()
            ) {
                JetpackInkDemoScreen(
                    colors = listOf(
                        Color.Blue, Color.Green, Color.Red, Color.Yellow
                    )
                )
            }
        }
    }
}
```

That's what we will be focusing on 😊



Experts

SnappMobile

```kotlin
val finishedStrokes = rememberSaveable(
    saver = with(SerializationHelper()) {
        Saver(
            save = { strokes ->
                ArrayList(serializeStrokes(strokes))
            },
            restore = { strokes ->
                mutableStateListOf<Stroke>().apply {
                    addAll(deserializeStrokes(strokes))
                }
            }
        )
    }
) { mutableStateListOf<Stroke>() }
```

Parameter passed to `DrawingSurface`.
The list contains all completed strokes.

No persistence, but we survive configuration changes 😎

Updated by the color and tool selectors

```
1  var currentColor by remember { mutableStateOf(colors.first()) }
2  var brushFamily by remember { mutableStateOf(BrushFamily.Pen) }
3  val brush = remember(currentColor, brushFamily) {
4      Brush.createWithColorIntArgb(
5          family = when (brushFamily) {
6              BrushFamily.Pen -> StockBrushes.pressurePenLatest
7              BrushFamily.Highlighter -> StockBrushes.highlighterLatest
8          },
9          colorIntArgb = when (brushFamily) {
10             BrushFamily.Pen -> currentColor.toArgb()
11             BrushFamily.Highlighter -> currentColor.copy(alpha = 0.4F).toArgb()
12         },
13         size = when (brushFamily) {
14             BrushFamily.Pen -> 5F
15             BrushFamily.Highlighter -> 55F
16         },
17         epsilon = 0.1F
18     )
19 }
```

```
enum class BrushFamily { Pen, Highlighter }
```

Parameter passed to `DrawingSurface`.
Brushes are essential for inking

```kotlin
@Composable
fun DrawingSurface(
    finishedStrokes: Collection<Stroke>,
    brush: Brush,
    modifier: Modifier = Modifier,
    addStrokes: (Collection<Stroke>) -> Unit
) {
    val canvasStrokeRenderer = remember { CanvasStrokeRenderer.create() }
    val latestBrush by rememberUpdatedState(brush)
    Box(modifier = modifier.clipToBounds()) {
        InProgressStrokes(
            defaultBrush = brush,
            nextBrush = { latestBrush },
            onStrokesFinished = addStrokes
        )
        FinishedStrokes(
            finishedStrokes = finishedStrokes,
            canvasStrokeRenderer = canvasStrokeRenderer
        )
    }
}
```

```kotlin
DrawingSurface(
    finishedStrokes = finishedStrokes,
    brush = brush,
) { strokes ->
    finishedStrokes.addAll( elements = strokes)
}
```

SnappMobile

```
val canvasStrokeRenderer = remember { CanvasStrokeRenderer.create() }
```

```
@Composable
fun FinishedStrokes(
    finishedStrokes: Collection<Stroke>,
    canvasStrokeRenderer: CanvasStrokeRenderer
) {
    val canvasTransform = remember { Matrix() }
    Canvas(modifier = Modifier.fillMaxSize()) {
        // This is a no-op as canvasTransform is an identity matrix.
        drawContext.canvas.nativeCanvas.concat(canvasTransform)
        finishedStrokes.forEach { stroke ->
            canvasStrokeRenderer.draw(
                stroke = stroke,
                canvas = drawContext.canvas.nativeCanvas,
                strokeToScreenTransform = canvasTransform
            )
        }
    }
}
```

Wrap up

- Works great, fun to use
- Some areas look and feel a little barebones
- Part of the philosophy
  - Get the basic parts stable
  - Add more goodies along the way

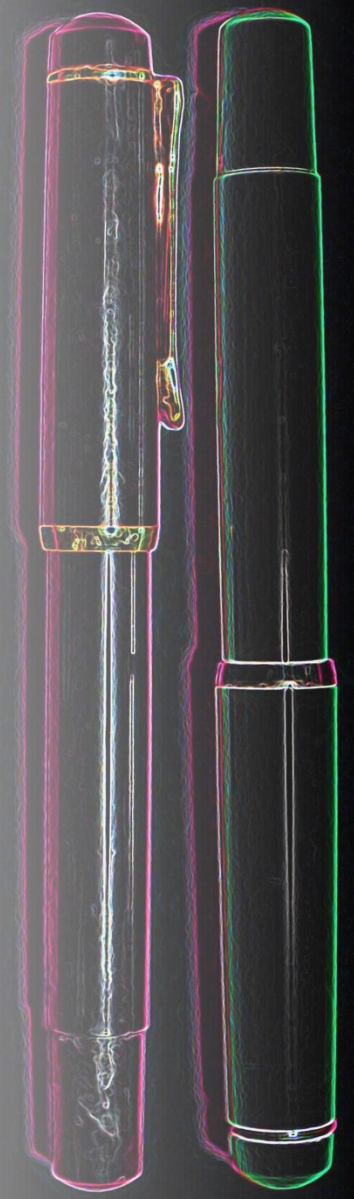


Photo Art by Monika Künneth

What to look at next

- Persistance and state preservation: basic loading and saving capabilities in the `storage` package
- Editing and erasing: support available through the `geometry` package
- Scrolling and zooming: nothing there, it’s entirely up to you

Photo Art by Monika Künneth

Experts

Resources

- Android Developers Blog post; https://android-developers.googleblog.com/2024/10/introducing-ink-api-jetpack-library.html
- API Reference https://developer.android.com/jetpack/androidx/releases/ink
- Add inking to your app with the Ink API https://developer.android.com/develop/ui/compose/touch-input/stylus-input/about-ink-api
- Ink API Compose by Nicos Nicolaou https://github.com/NicosNicolaou16/Ink_Api_Compose

Thank you!

@tkuenneth

@tkuenneth

@tkuenneth.dev

https://github.com/tkuenneth/JetpackInkDemo

Experts

SnappMobile