Calc Tiled Rendering in LibreOffice Online

Ashod Nakashian
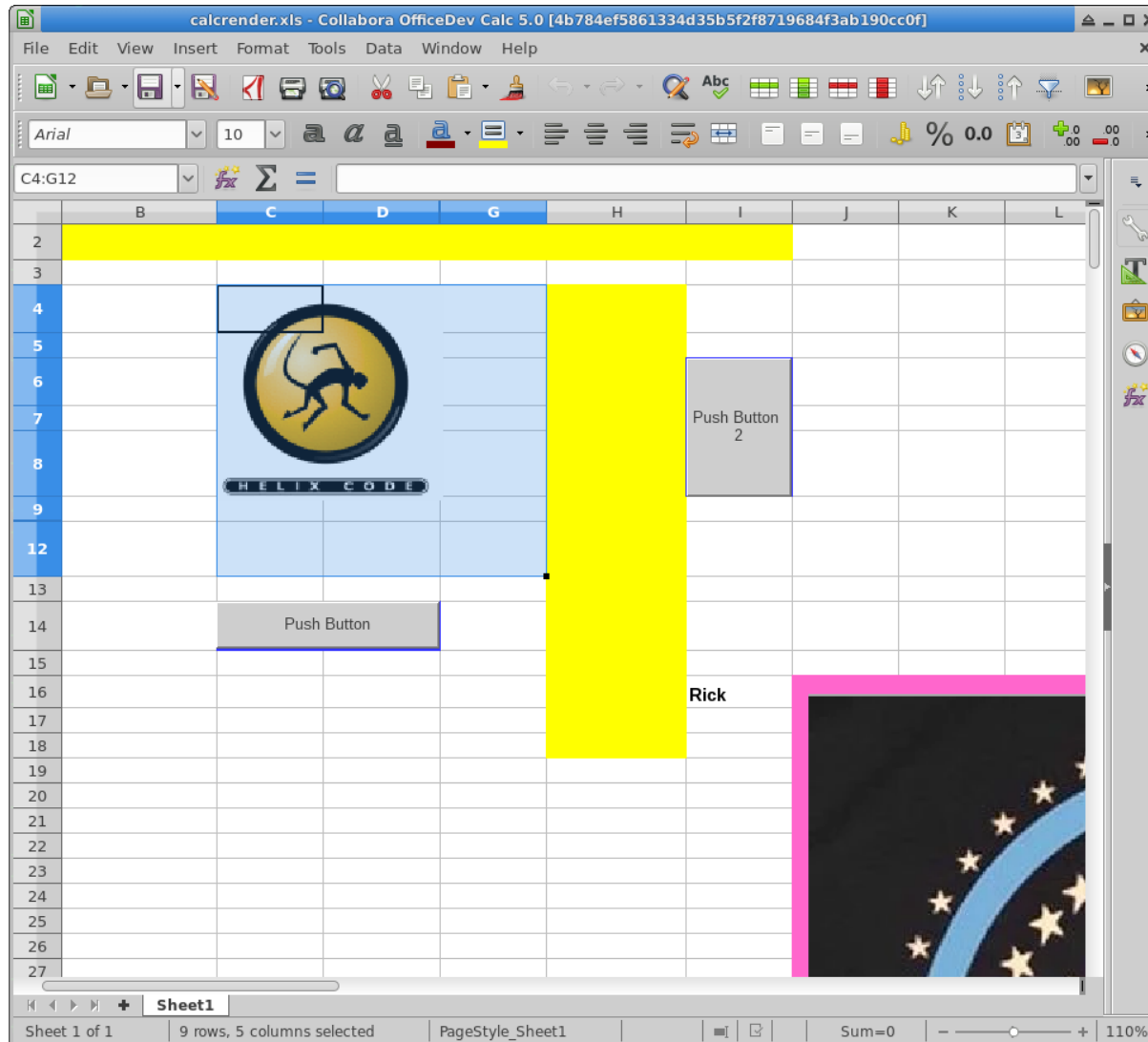
Collabora Productivity

Overview

- **Rendering: Desktop vs Online**
  - The Performance Gap
  - Why?
  - Fix it → Surprise!
- What's the problem?
  - What's the solution?
  - You are your enemy
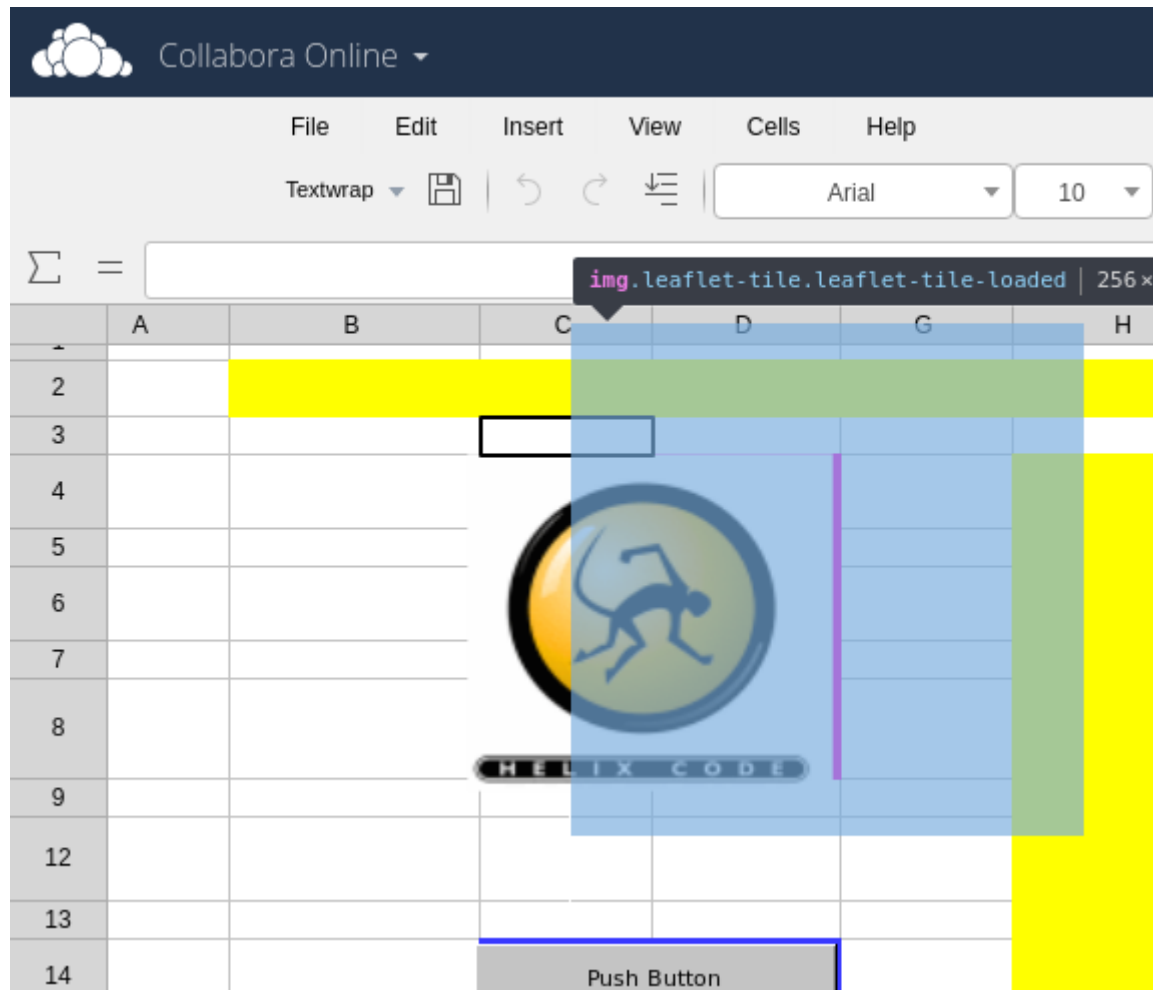 - What's the *real* solution?
- Results
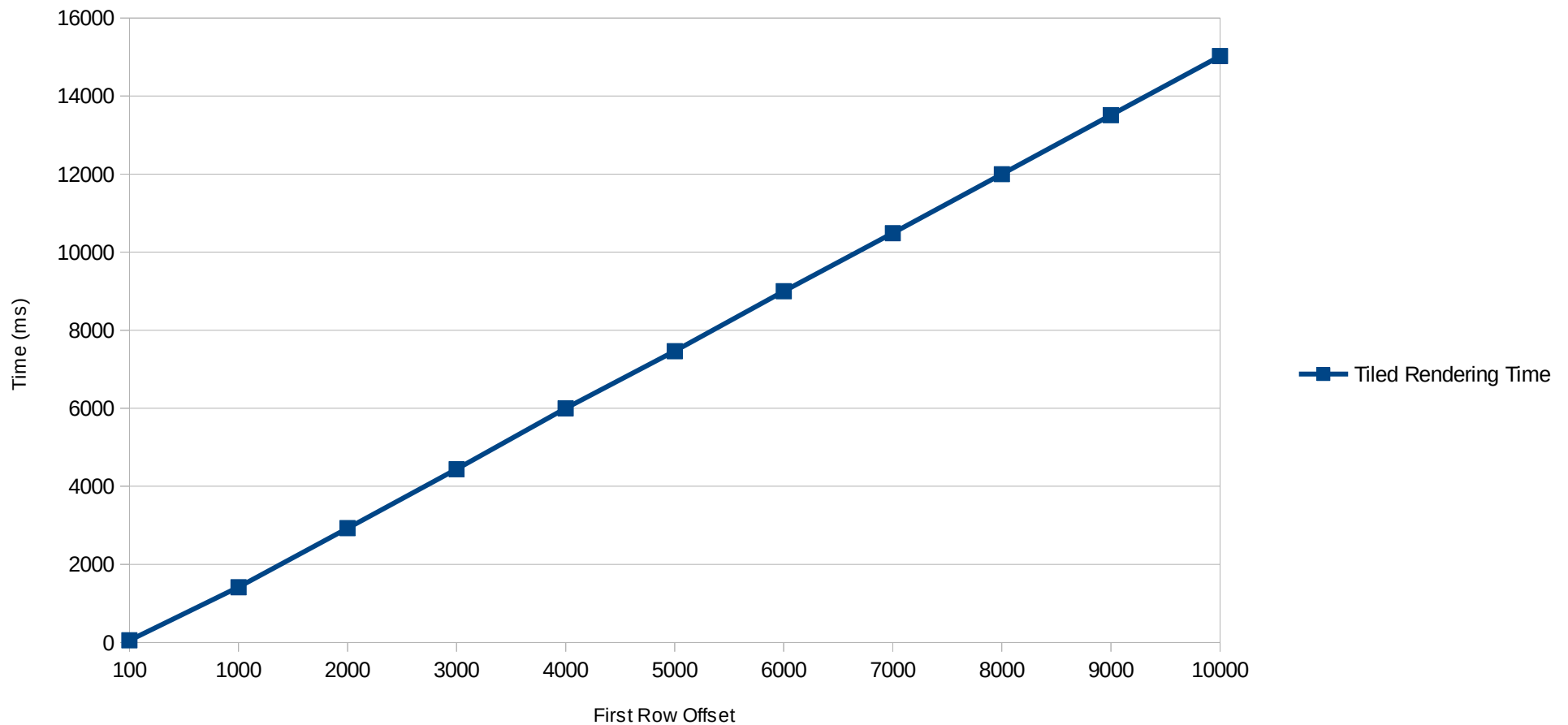
Rendering: Desktop vs Online

Rendering in Desktop

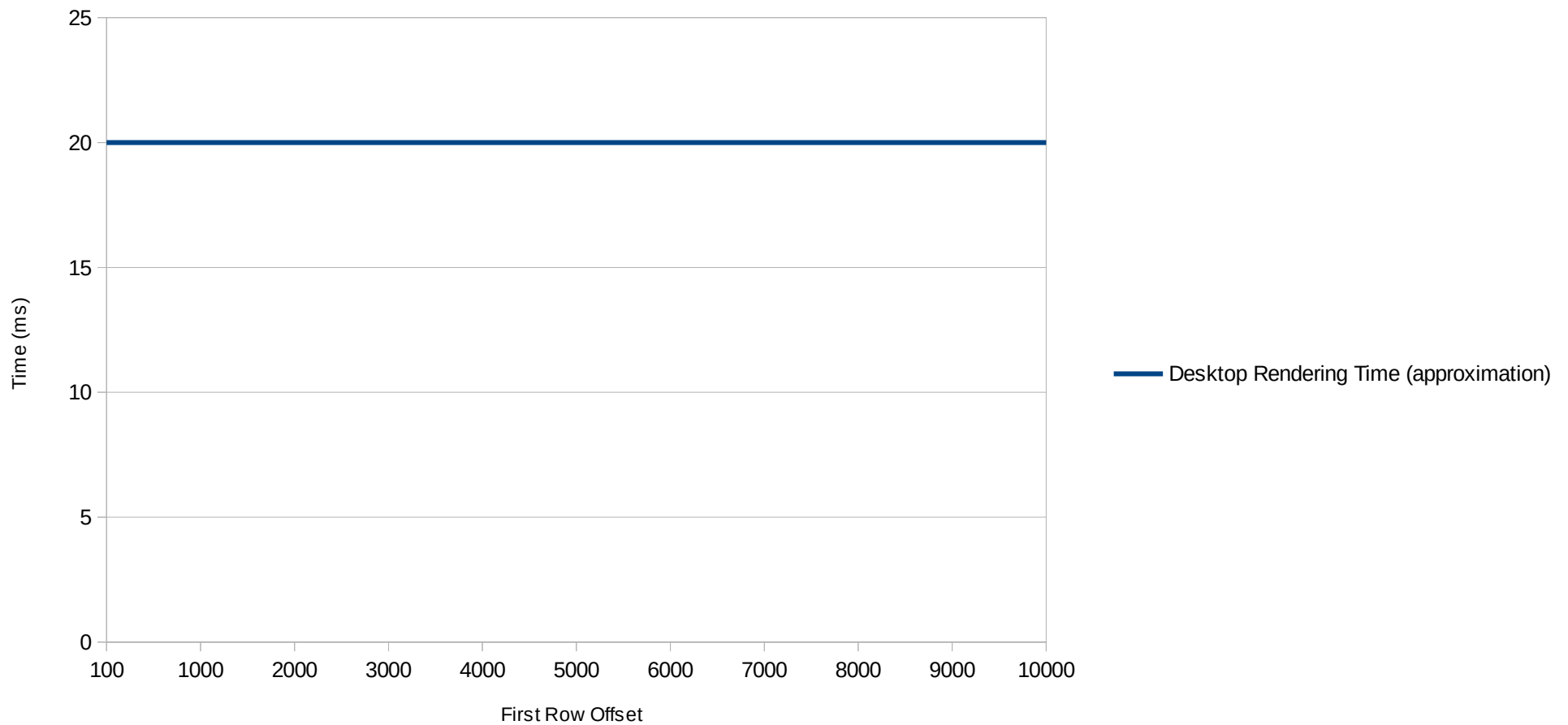Collabora Productivity

Rendering in Online

Rendering Performance in Online

Tiled rendering time proportional to Row offset!

Collabora Productivity

Rendering Performance in Desktop
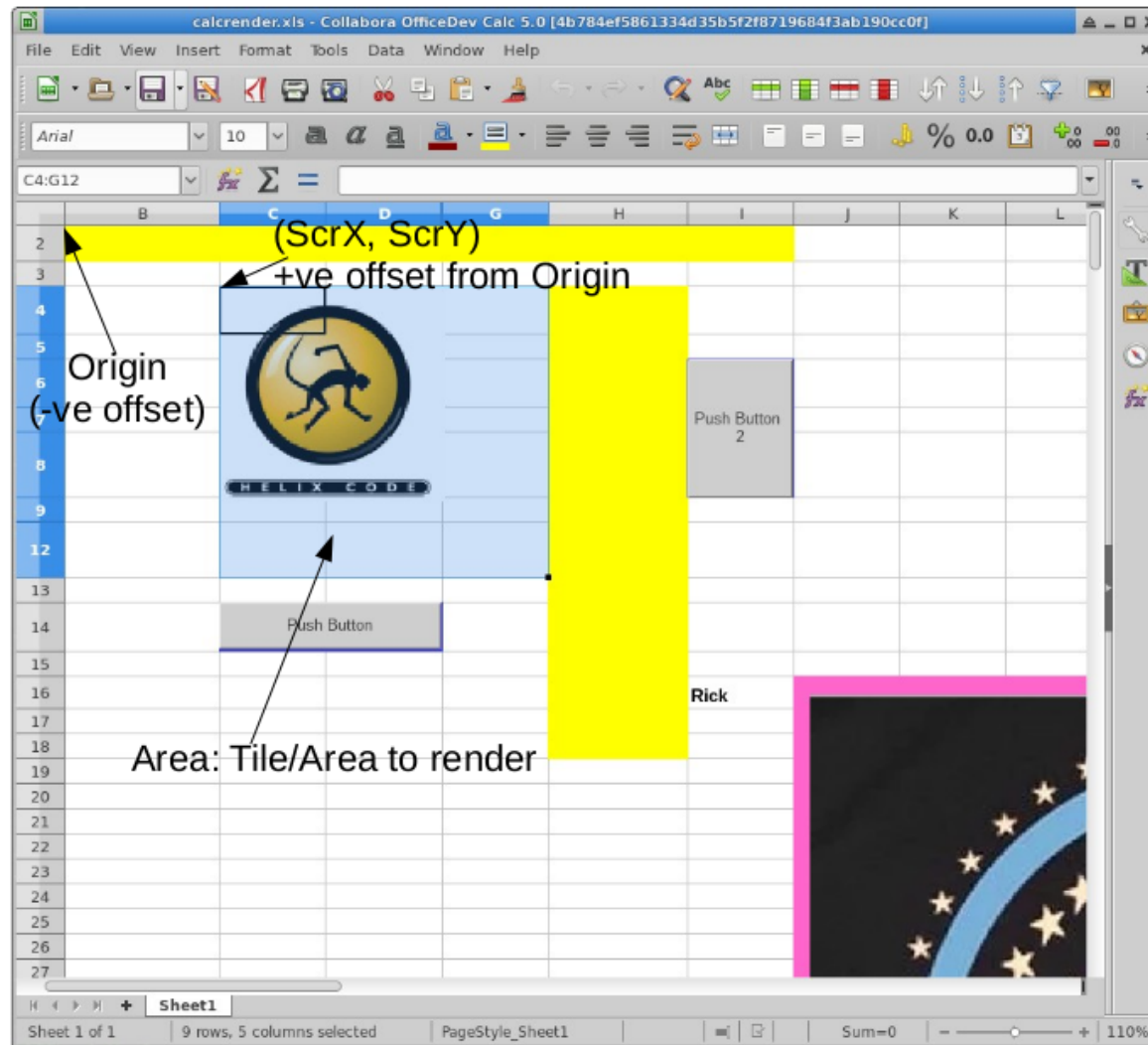
Desktop rendering time (approximation)

Why is Desktop Rendering Fast?

- Scrolling aligns to column-boundary
- Renders only the visible columns/rows
- Does not suffer special cases (more on this)

Rendering in Desktop: Anatomy

Why is Tiled Rendering Slow?

One Reason: Renders everything from (0, 0) to bottom-right corner of the tile (hence, the linear-progression graph)

Make it Faster

- Render only the bounding cells
- Calculate two offsets:
  1) The offset to the first visible row/column
  2) The offset of the top-left *tile* pixel from the top-left cell's origin
- Render only the bounding cells, given the calculated offsets

Initial Results: Fast, but...

Button

Missing Button

Initial Results: Fast, but...

Missing Image

The Problem

What's going on?

- Embedded images and Buttons are rendered in a different coordinate system than Text, Backgrounds, etc.

- Unlike Desktop, Online must ultimately render everything on a virtual device and in Pixel coordinates
    - Online path changes the coordinate system to Pixel explicitly

The Magic that is MapMode

- Defines the coordinate system
- Specifies the reference point (origin) of the coordinate system
- Specifies the scale of the coordinate system
- Automagically converts between logical and physical coordinates

The Trouble with MapMode

- In an ideal world all coordinates would be in Platonic Form (i.e. in the most accurate form of representation—perfect)
- In an ideal *real* world there would be one logical coordinate system: the physical one (i.e. 1-to-1 mapping)
- In practice, LO uses many coordinate systems (MapUnits): **TWIPS**, **100TH_MM**, **PIXEL**, etc. and needs to convert from logical to physical, and vice versa

It Gets Worse

- MapMode can be enabled/disabled by a flag
- Code selectively saves and restores MapMode around function calls.
  - This is not perfect or bug free, of course
- Some Code expects implicit MapMode, others set it explicitly themselves
  - Fragile: breaks with the least disruption

What's the Solution?

- Correct the MapModes of each rendering layer/step **_explicitly_** to map correctly to the physical coordinates (i.e. the pixels in the virtual device)
    - Do this for Tiled Rendering only (i.e. don't break Desktop Rendering)

The Algorithm

- Log the origin and scale
  - Compare with Desktop logs
  - Figure out from the scale, the conversion factor that needs backing-out
- Binary search where MapMode changes
- Add MapMode with correct origin/scale
- Repeat as necessary

New Results

Push Button

Rick

Seems to work: Great!

Except...

Scaled and shifted graph: FAIL!

Name	Value
A	1
B	2
C	3
D	4

same	same	same	same	same	same	same

Rich *text* content

You are your enemy

- Every time we change/insert a MapMode we potentially affect all subsequent rendering
  - No complete list of all possible rendering layers/elements etc. and the respective MapMode each expects
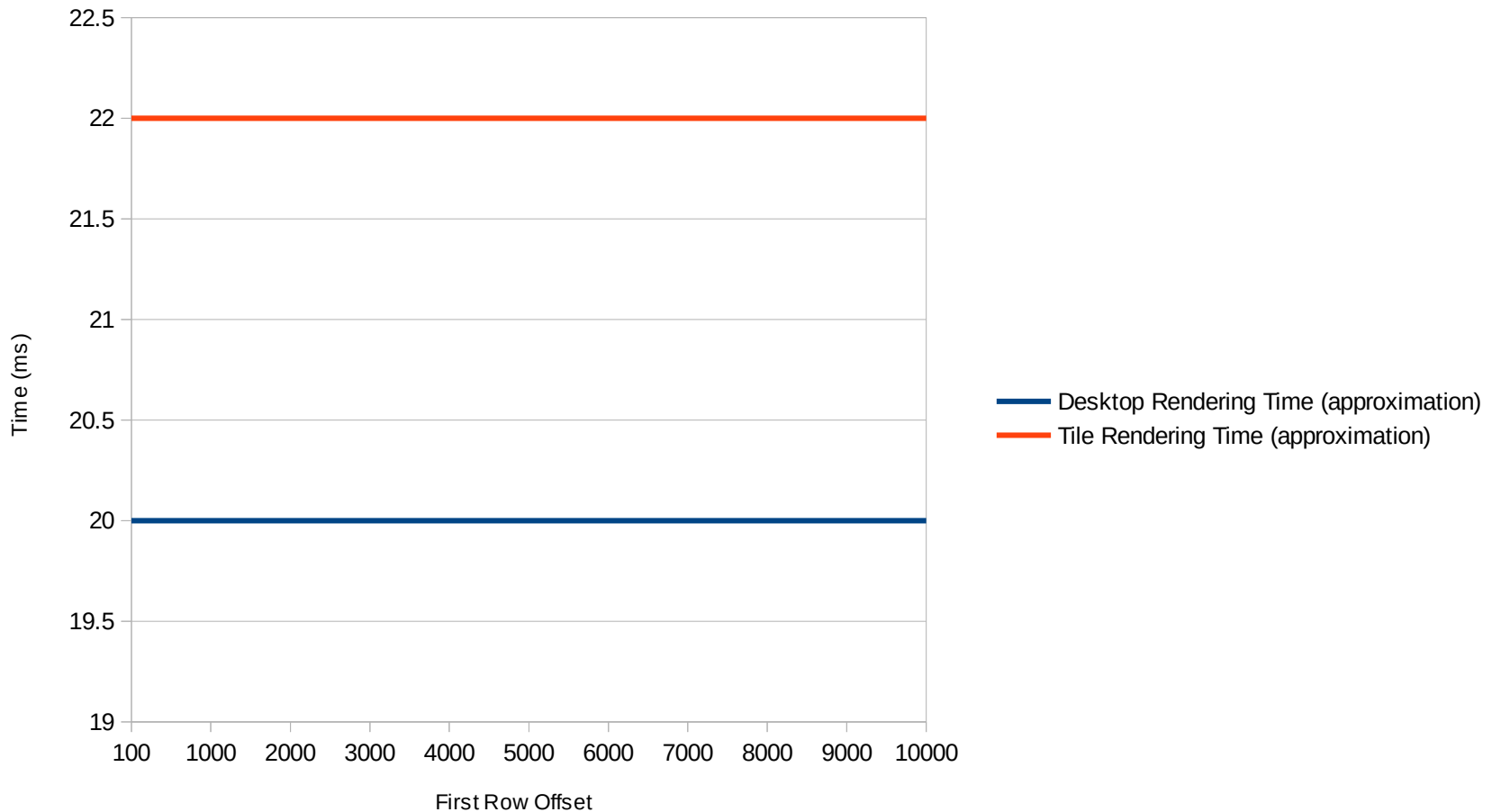- Explicitly adding MapMode is a vicious cycle

Rinse and Repeat

… until all known cases are fixed, and rendering is both correct and fast.

Rendering Performance
Tiled & Desktop

Tiled & Desktop rendering time (approximation)

Ideally…

… MapMode will be set once on the device and forgotten.

In Practice…

… changing the rendering code to always use 100TH_MM MapMode proved exceedingly, and prohibitively, hard and expensive.

But...

… there must be someone Smart Enough^{TM}, out there, who can do it.

Thank You

- <Your Question Here>
-