Implementation of a Vulkan-based renderer in Processing

AUTHOR & PRESENTER:
Téo Taylor

Speeding up the **Processing framework** with a new **Vulkan renderer.**

BACKGROUND:

Processing is a creative coding framework that makes it easy to create stunning hardware-accelerated visuals.
However, it uses **OpenGL which is old and slow**. It has been superseded by **Vulkan which is modern and fast**.

This study implements Vulkan into Processing and compares its performance against OpenGL.

METHODS

1. Write a light OpenGL-to-Vulkan translation layer, specifically optimised for Processing.
2. Write a set of tests using the Processing Language.
3. Measure runtime over x number of frames with the OpenGL and Vulkan renderers.
4. Collect Intel VTune profiling results with the OpenGL and Vulkan renderer to identify performance improvements/declines reasons.

RESULTS

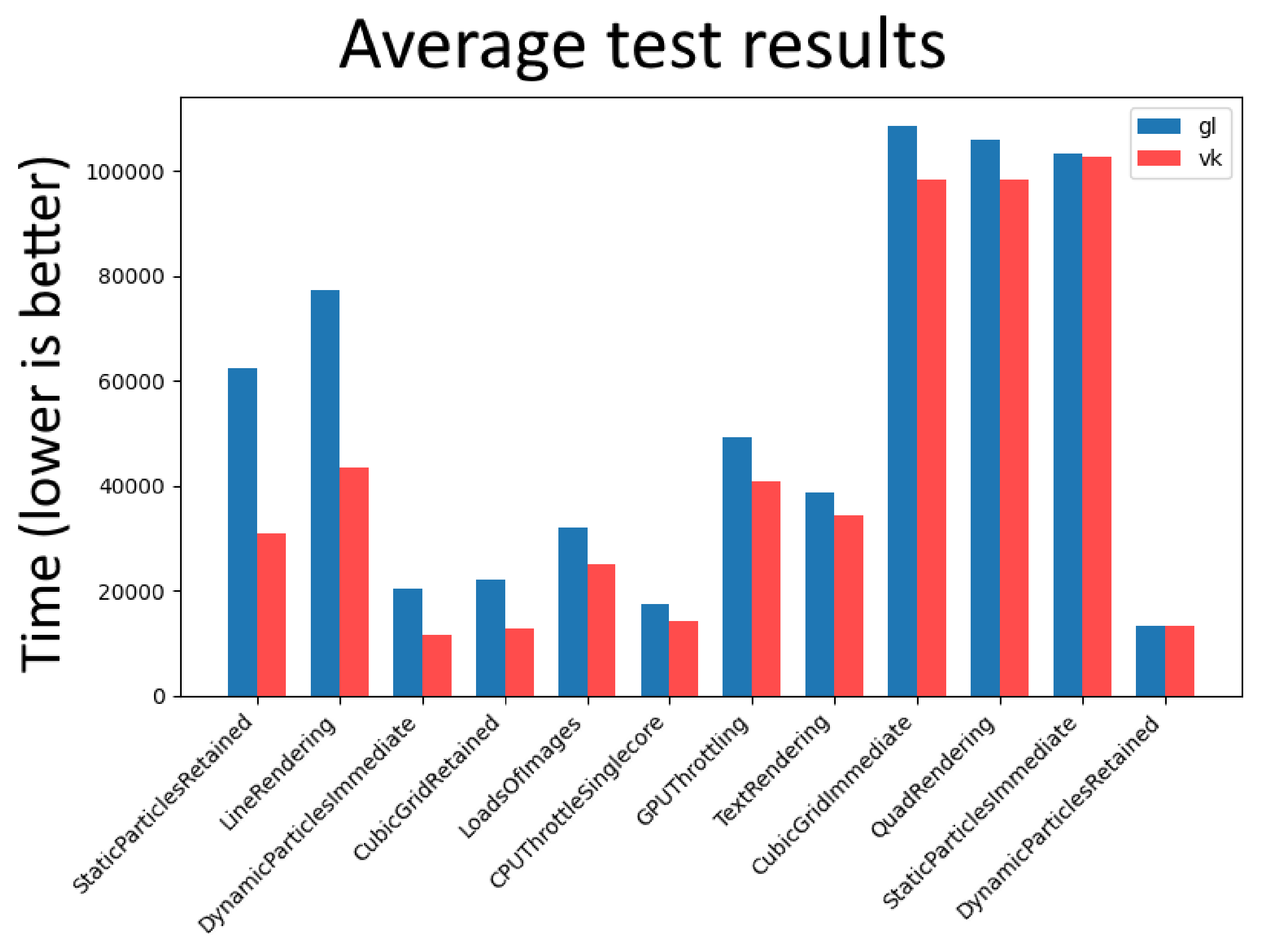
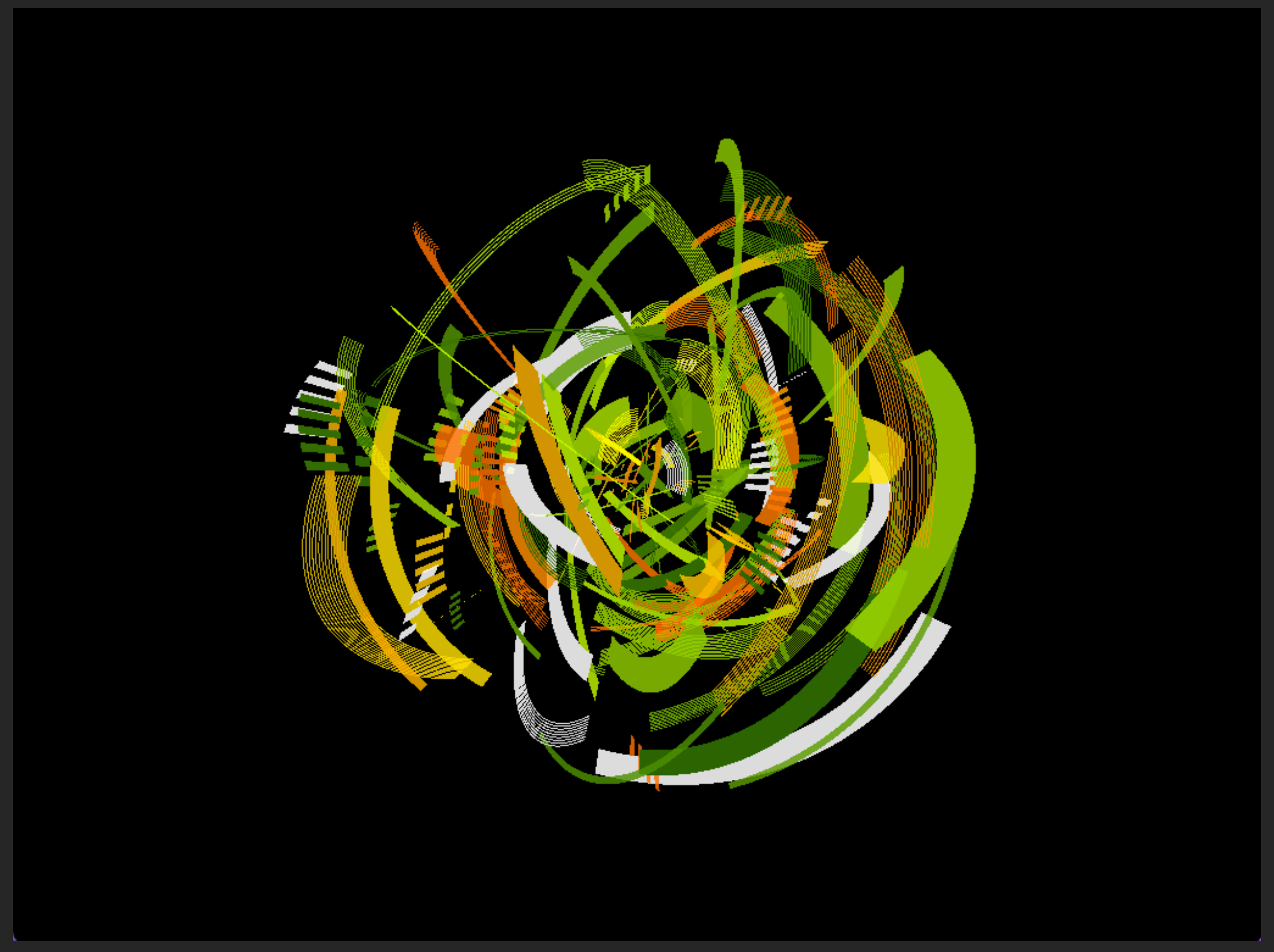
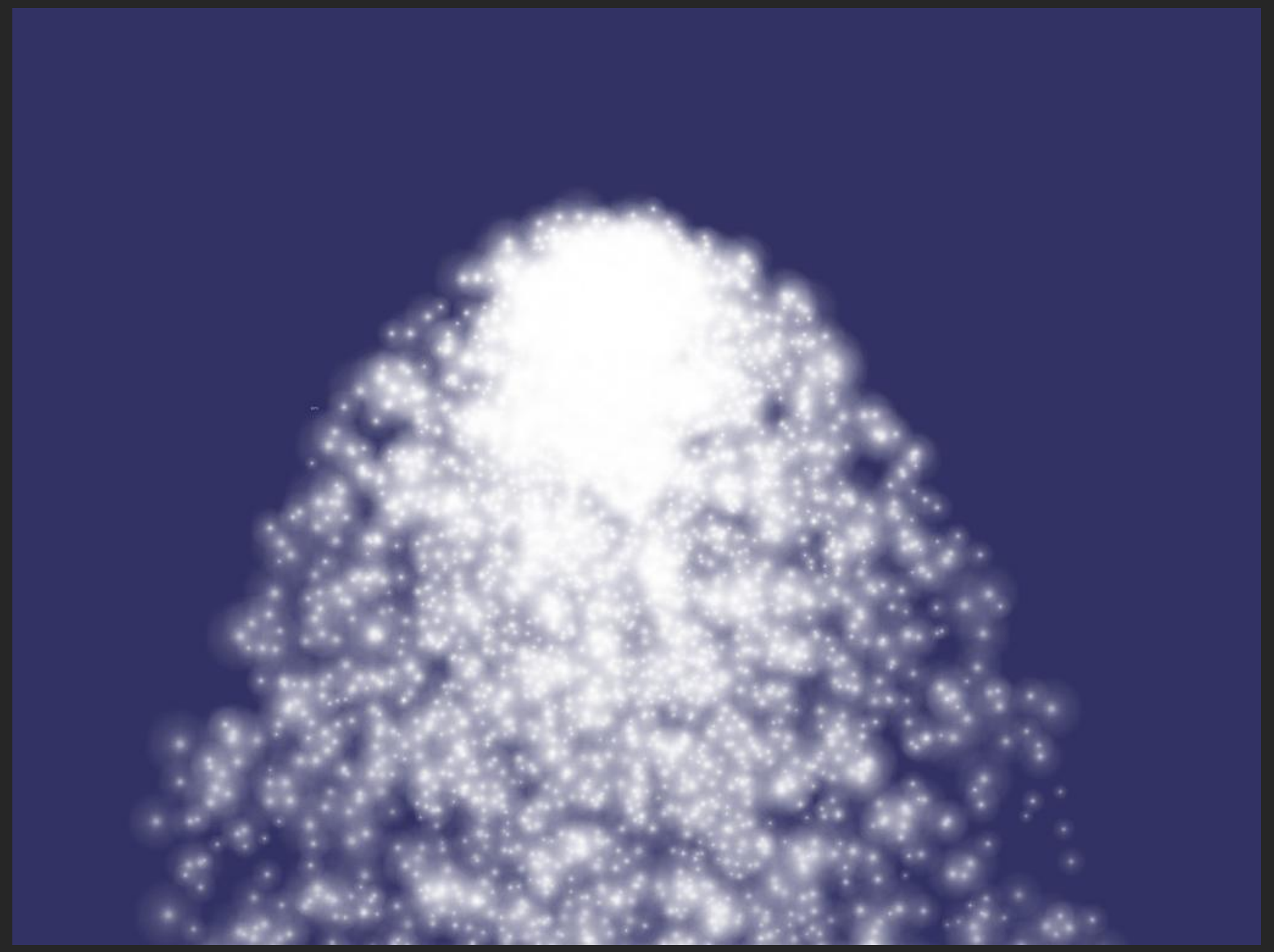
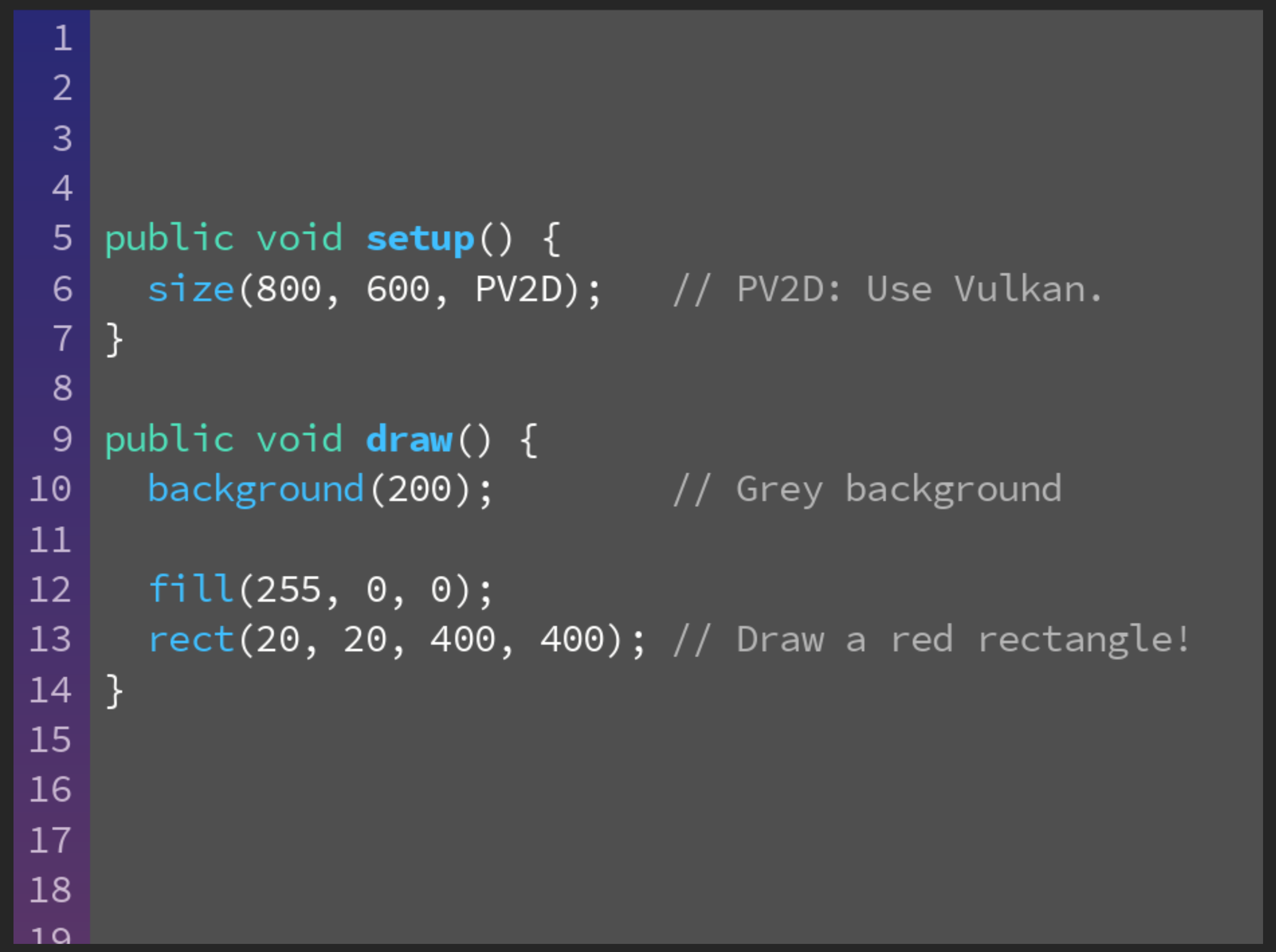


***Figure 1**: The average mean of the time data in all tests.*

On average, Vulkan is **~20% faster** than OpenGL, and **up to 3.5 times faster** depending on the sketch.



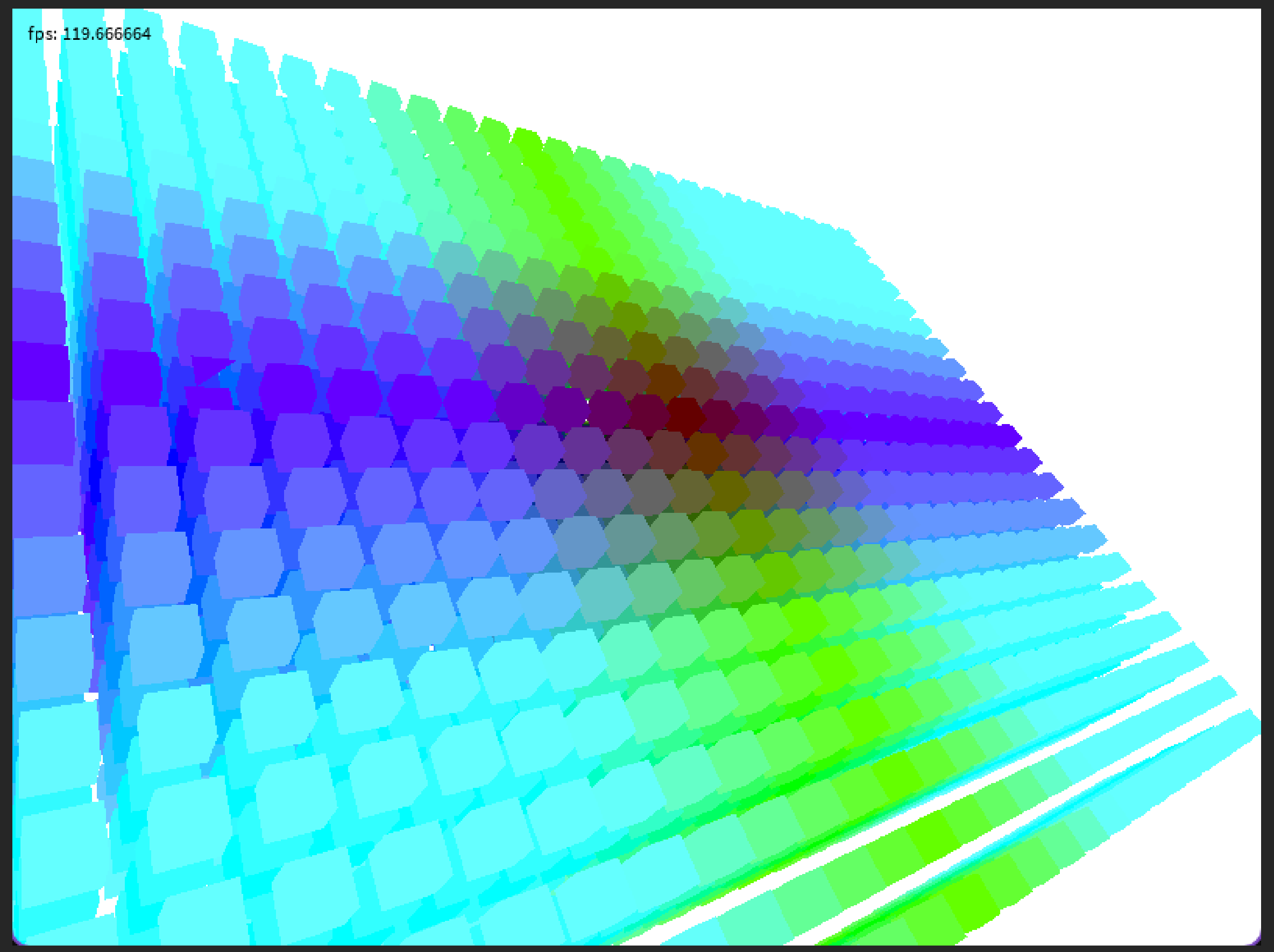




```
public void setup() {
  size(800, 600, PV2D);   // PV2D: Use Vulkan.
}

public void draw() {
  background(200);        // Grey background

  fill(255, 0, 0);
  rect(20, 20, 400, 400); // Draw a red rectangle!
}
```



Scan to
download the **full paper**

Line Rendering test – Screenshots, graphs, and Intel VTune profile results.

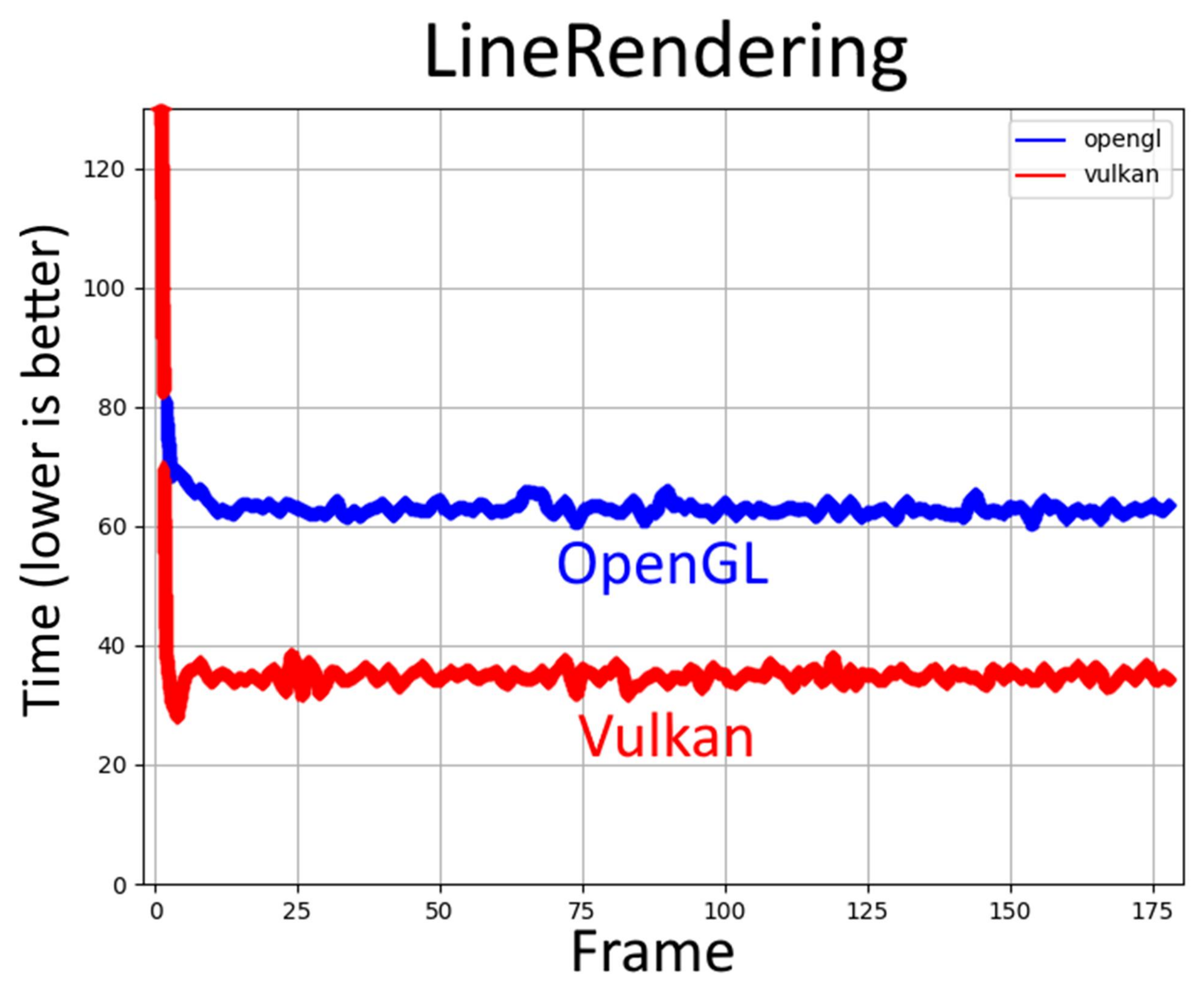


***Figure 2**: Performance graph of the Line Rendering test. Times are in milliseconds.*

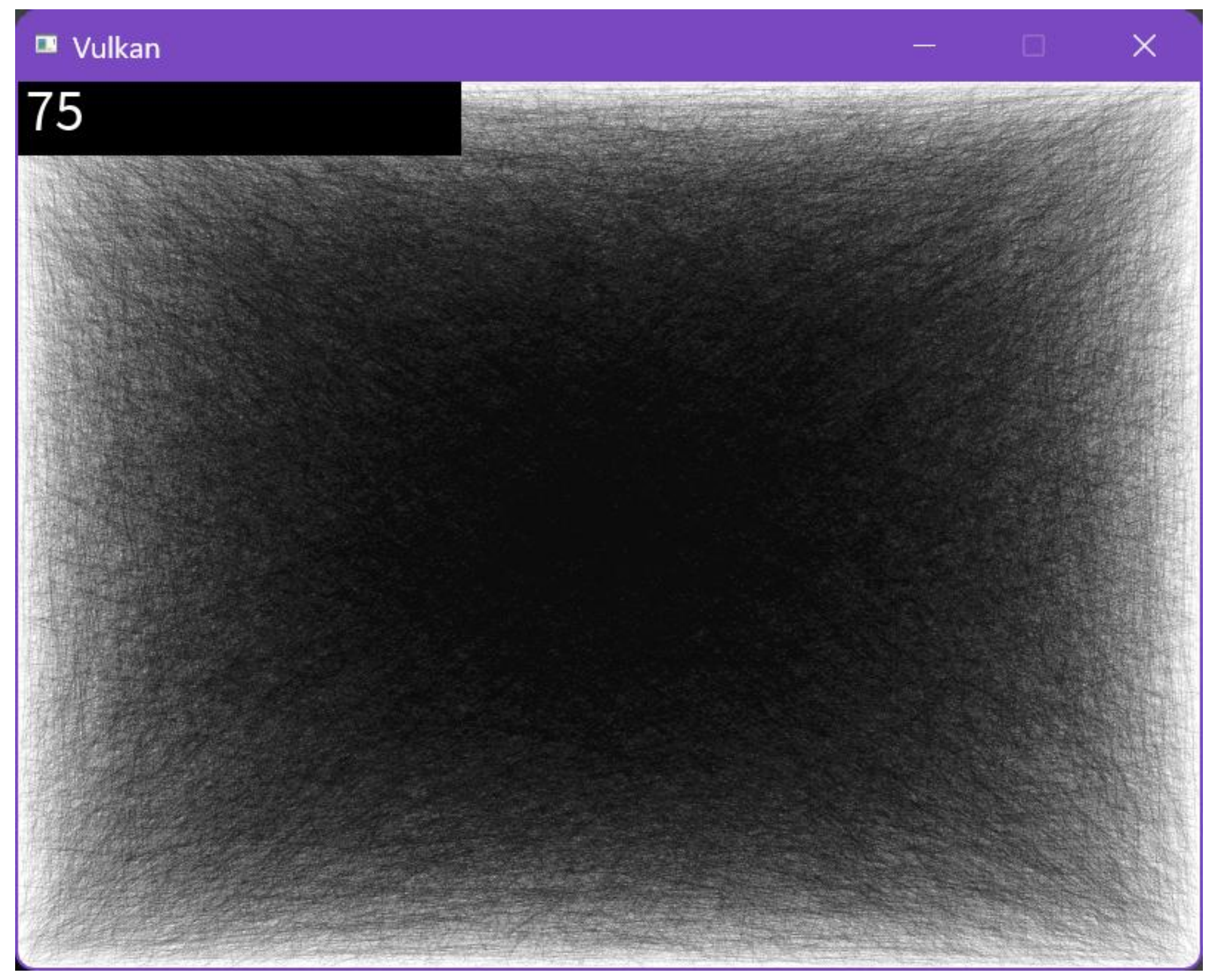


***Figure 3**: Screenshot of the Line Rendering test in Vulkan.*

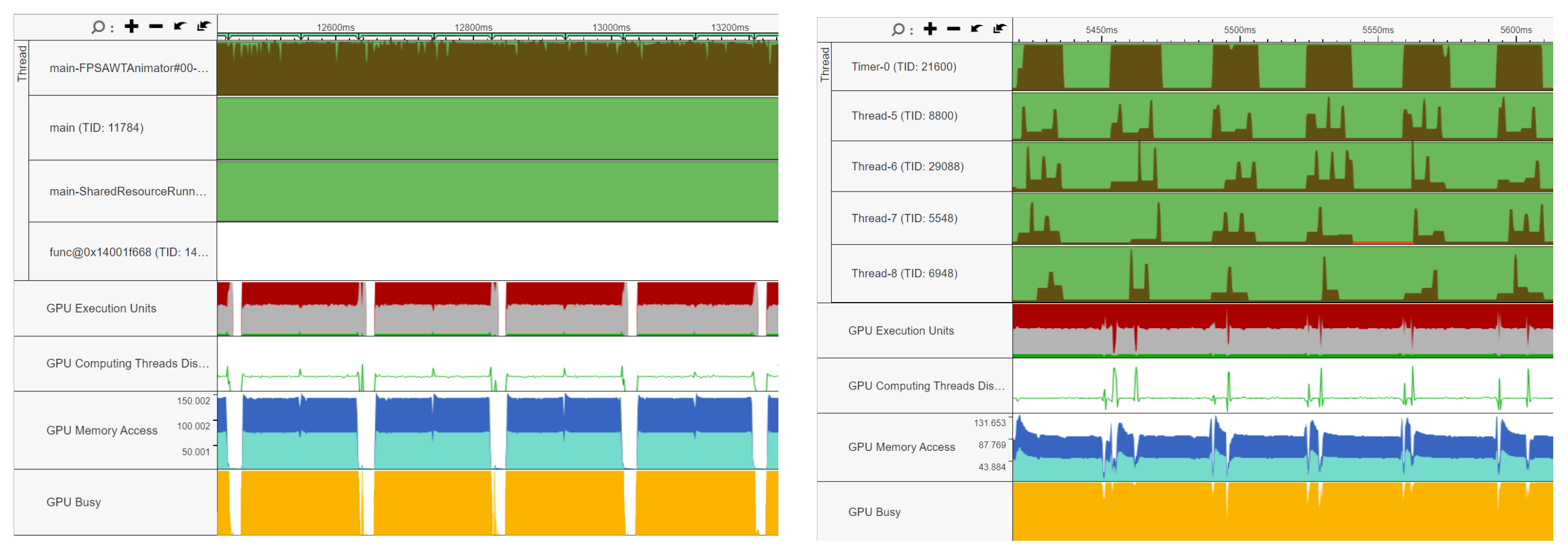


***Figure 4**: Intel VTune profile results for the OpenGL renderer (left) and the Vulkan renderer (right).*

HERIOT WATT UNIVERSITY