WebMantle

Modern 3D API for the Web

Dzmitry Malyshau
kvark@mozilla.com

But… Why?

- GL is outdated, no longer maps to HW nicely
- GLSL is inferior to SPIR
- Performance:
  - parallel construction of the command buffers
  - re-use of them with different parameters
  - pipeline state objects → less CPU driver work
  - faster shader loading with SPIR

Design

- Concepts:
  - Command Buffers, ~~Encoders~~ *(metal)*
  - Pipeline State Objects *(metal/vulkan)*
  - Texture/Buffer Views *(d3d11/vulkan)*
  - Descriptor Sets *(vulkan)*
- SPIR-V shaders *(vulkan)*
- Left behind:
  - Resource/Memory heaps
  - Explicit synchronization
  - Explicit state transitions

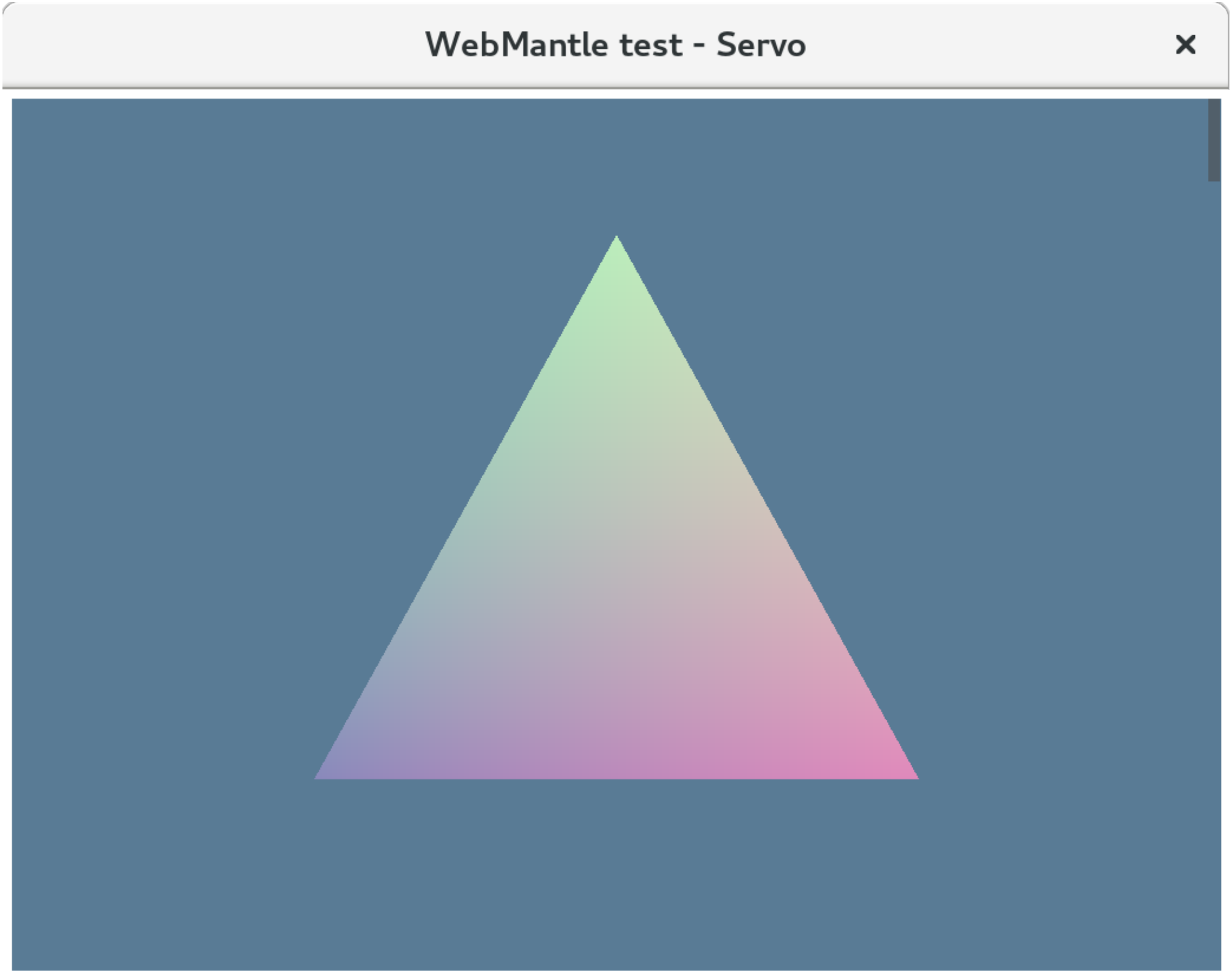
- https://www.khronos.org/webgl/public-mailing-list/archives/1508/msg00023.php

Prototype Implementation

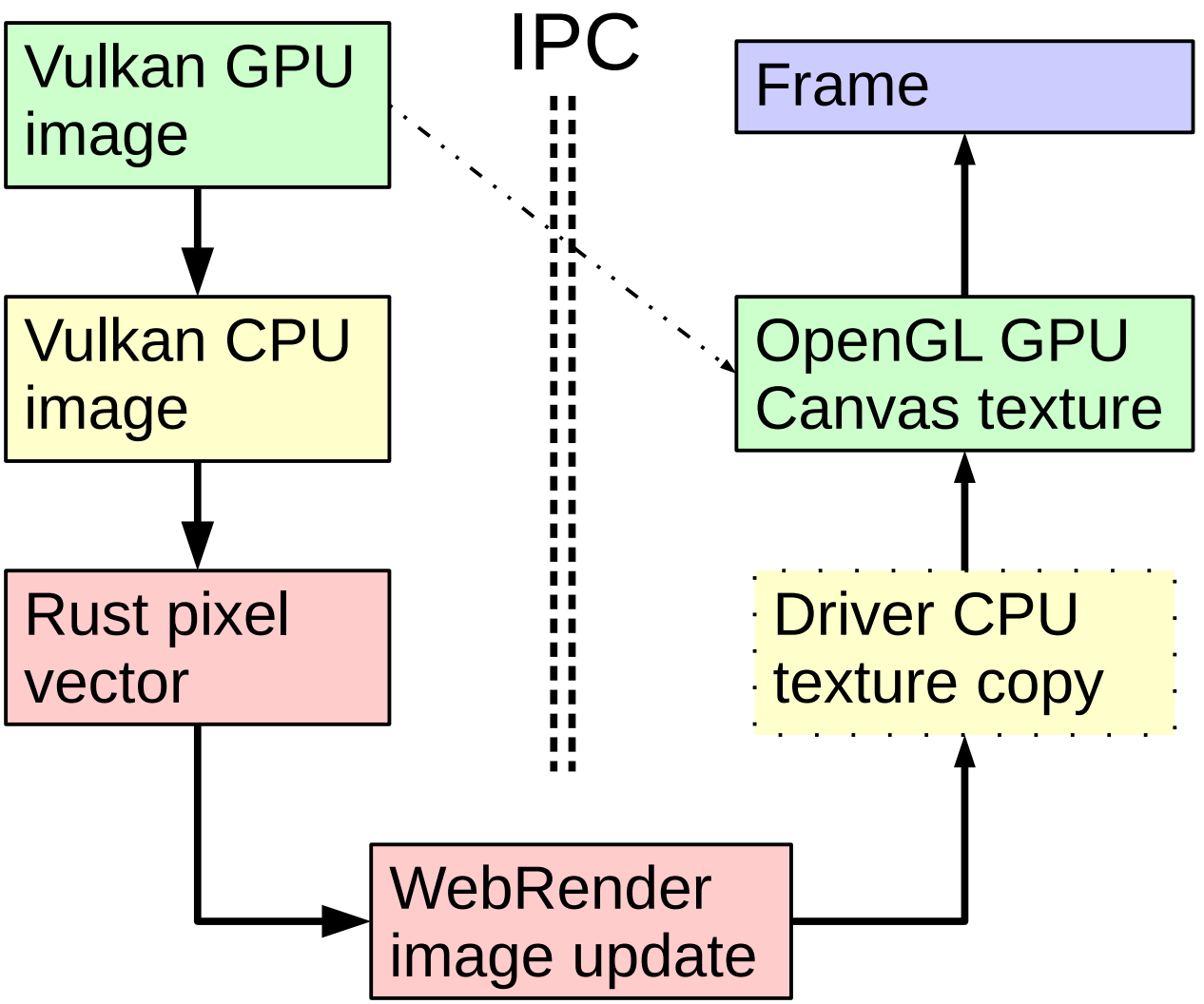
- Built into Servo, made with Rust
- Vulkan backend:
  - Swap chain (read-back)
  - Auto-threading behind IPC
  - Trackers for :
    - command buffers
    - command encoder threads
    - resource layouts
  - Render pass creation, clearing, and drawing
  - Basic pipeline creation with GLSL shaders

Demo JS code

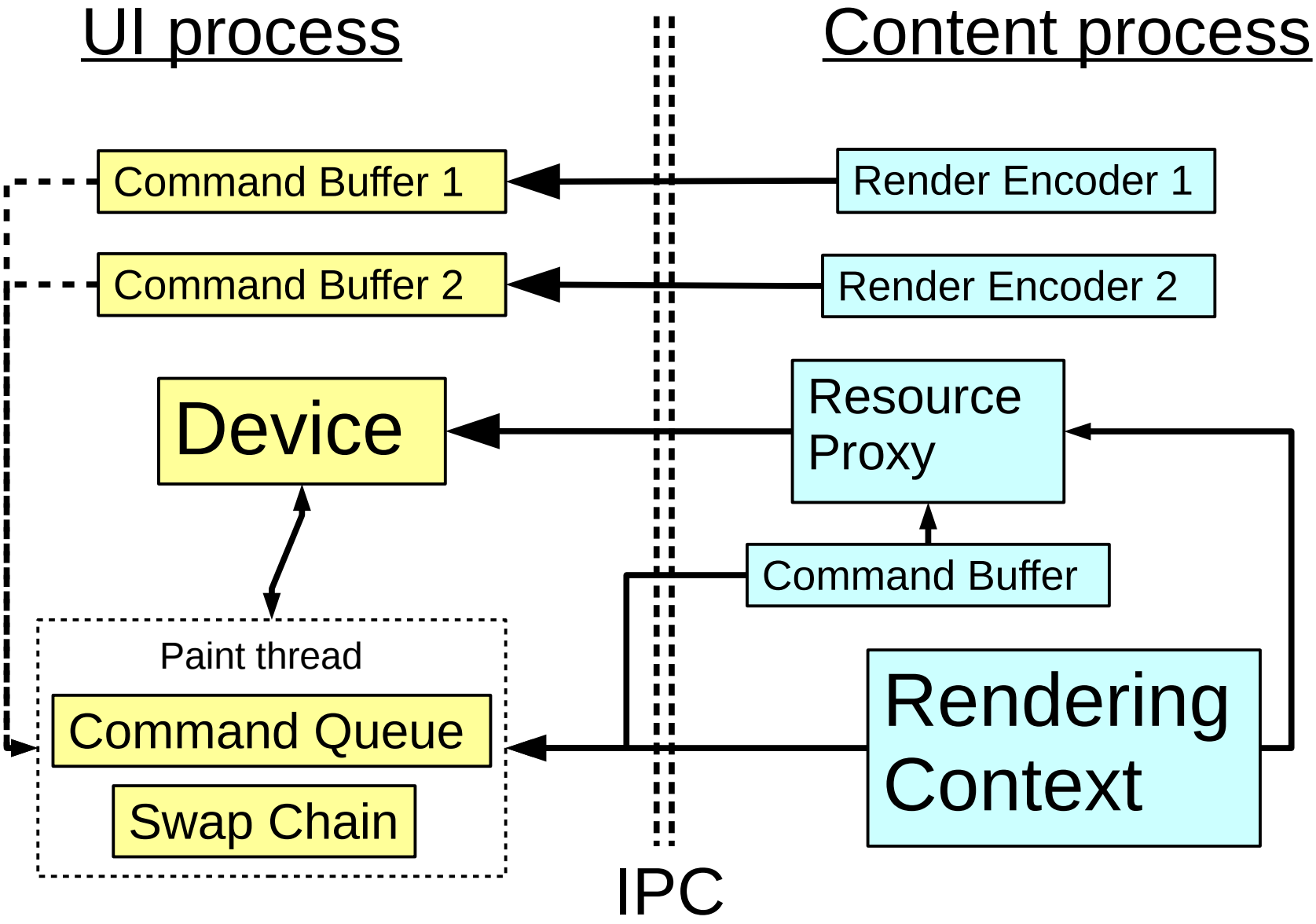
```
var target = context.nextFrameTarget();
var commandBuffer = context.makeCommandBuffer();

var renderEncoder = commandBuffer.makeRenderCommandEncoder({
      color0: { view: target, clear: [0.1, 0.2, 0.3, 1.0] },
});
renderEncoder.setRenderPipelineState(pipeline);
renderEncoder.drawPrimitives(0, 3, 1);
renderEncoder.endEncoding();

commandBuffer.commit();
context.endFrame();
```


WebMantle test - Servo

Read-back Presenting



Auto Threading



Still Undecided

- Name
- Indirect draws
- Sub Passes
- Command Encoders
- Command Buffer chaining
- Multiple Queues/Devices