Technical Report: Day-5R VRAM Governor (with Real VRAM Commitment)

1. Background & Motivation

Modern GPUs have limited VRAM. If an application suddenly exceeds available memory, the driver can stall or evict resources, causing stutter. To avoid this, our project builds a **VRAM Governor** — a control loop that monitors free VRAM and **sacrifices less important visual quality first** to stay within safe limits.

- **Day-4**: First working prototype. It consumed VRAM using dummy textures ("pads") and the GPU driver reported actual free memory correctly on both desktop and laptop.

- **Day-5**: Added **priority-aware** and **spike-aware** logic. But VRAM consumption didn't show up on some drivers, causing confusion.

- **Day-5R**: Fixed version. Now pads are **touched** to force real VRAM commitment, and the governor automatically falls back if telemetry is frozen.

2. Pads — Simulated VRAM Load

What is a pad?

A **pad** is a giant dummy GPU texture, created solely to **artificially consume VRAM**.
- Size: 8192 × 8192, RGBA8 format
- Base memory: 256 MiB
- With full mipmaps: 341 MiB

Each time you press **B**, one pad is allocated → free VRAM drops.
Pressing **Shift+B** frees one pad → VRAM rises.

Why we must "touch" them

- **Problem (Day-5):** `glTexStorage2D` reserves space, but doesn't always back it with physical memory. The driver may keep it "virtual" until written to. Result: telemetry didn't change.

- **Solution (Day-5R):** Each pad is:
  1. Attached to an FBO.

  2. Cleared once (`glClear`).

  3. Passed through mipmap generation (`glGenerateMipmap`).

This forces the driver to **commit VRAM** physically, so telemetry (e.g. NVX extension) really decreases.

Analogy: booking hotel rooms vs actually checking in. Day-5 only reserved rooms; Day-5R checks in guests, so occupancy shows up.

3. Telemetry vs. Fallback

Telemetry

- Uses GPU driver extensions:
  - `GL_NVX_gpu_memory_info` (NVIDIA)
  - `GL_ATI_meminfo` (AMD)
- Reports free VRAM in KB.
- Works well on desktops. On laptops, often broken (returns a fixed number).

Fallback

- Pure software counter:

  `freeMB = baseline  (#pads × 256)`
- Always predictable, even if the driver lies.

Watchdog auto-switch

- After each pad allocation, we check if telemetry dropped by 128 MB.
- If not for two allocations → **assume telemetry frozen** → switch to fallback automatically.
- This is why you don't need to press **C** manually anymore.

4. Governor Control Logic

The governor runs every ~0.25 s. Its goal: keep ~**1024 MB free VRAM** with a ±128 MB band.

Inputs

- Current freeMB
- Change from last frame (`delta`)

Behavior

1. **Spike detection ("tourniquet")**
   - If `delta  256 MB` (sudden drop), call `spike()`.
   - This adds a **large increment** to `biasLow`.
   - Analogy: If the patient is bleeding out, apply a tourniquet — sacrifice the least important visuals immediately.
2. **Hysteresis band (stability)**
   - Below 896 MB → escalate blur gradually.
   - Above 1152 MB → de-escalate (sharpen) gradually.
   - Between 896–1152 MB → no change (prevents flicker).
3. **Priority ladder**
   - Escalation (blurring): **Low → Normal → High**
   - Recovery (sharpening): **High → Normal → Low**
4. **Bias increments**
   - `stepGradual` (e.g. +0.5) for slow changes.
   - `stepSpike` (e.g. +1.25) for sudden changes.
   - Clamped to `[0 ... 8]` per panel.

5. Bias Variables vs. Shader `uBias`

Bias variables (CPU side)

- `biasLow`, `biasNorm`, `biasHigh`
- Stored in the governor.
- Represent "how blurred each priority class should be."
- Updated every ~0.25 s depending on VRAM state.

Shader uniform (`uBias`)

- Passed once per draw call:

```
drawPanel(..., gGov.biasLow,  gSceneTex);   // Left
drawPanel(..., gGov.biasNorm, gSceneTex);  // Center
drawPanel(..., gGov.biasHigh, gSceneTex);  // Right
```

- The fragment shader uses it in:

```
vec3 c = texture(uTex, vUV, uBias).rgb;
```

- This forces mip biasing → higher bias = blurrier mip level.

Connection

- Governor decides new bias values →
- Sent into `uBias` per panel →
- Shader samples blurrier mip levels →
- Panels blur in the intended order.

6. Visual Result

- **Press B**: VRAM drops. First the left panel blurs, then center, then right.
- **Press Shift+B**: VRAM rises. Right panel sharpens first, then center, then left.
- **Press]**: All biases nudged up → all panels blur instantly (debug check).
- **Press R**: Reset all pads & biases.

7. Day-4 vs Day-5 Confusion Explained

- **Day-4:** Pads were touched implicitly (e.g. uploaded data). Driver committed VRAM, telemetry moved → everything "looked real."
- **Day-5:** Pads only reserved, never touched. Driver didn't commit VRAM → telemetry flatlined → no blur triggered until fallback.
- **Day-5R:** Fixes this. We explicitly clear + mipmap each pad, so telemetry behaves like Day-4 again. If driver still lies, fallback auto-takes over.

8. Key Terms You Asked About

- **Pad**: A large dummy texture (~256–341 MiB) used to simulate VRAM load.
- **Touching a pad**: Writing once into the texture (clear/upload/mipmap) to force VRAM commitment.
- **Immediate tourniquet**: When freeMB spikes downward suddenly, immediately blur Low-priority content by a bigger step.

- **Priority ladder**: Ordered escalation (Low→Normal→High) and ordered recovery (High→Normal→Low).
- **biasLow/biasNorm/biasHigh**: CPU-side state variables for each class.
- **uBias**: Per-draw shader uniform set from those variables.
- **Telemetry**: GPU driver-reported free VRAM.
- **Fallback**: Software counter that pretends each pad is exactly 256 MB.

9. Analogy Recap

- Pads = hotel rooms. Reserving (`glTexStorage2D`) actually occupied. Clearing/mipmap = check-in.
- Spike response = tourniquet. If blood loss is sudden, act instantly on least critical part.
- Priority = rolling blackouts. Cut least essential neighborhoods first, reconnect critical ones first.

10. Tuning Guide

- **Target freeMB (1024)**: Raise to keep more cushion, lower to push visuals harder.
- **Band ±128**: Larger = more stable, smaller = more twitchy.
- **Spike threshold (256)**: Lower = more sensitive, higher = ignore small jumps.
- **Step sizes**: Bigger = obvious blur jumps, smaller = subtle.
- **Bias max (8)**: How blurry you allow panels to get.

Conclusion

Day-5R is a working prototype of a VRAM governor that:
1. Uses **real VRAM consumption** (pads touched to force commitment).
2. Reads telemetry when available, falls back when frozen.
3. Applies **priority-aware** blur adjustments: Low first, High last.
4. Reacts to sudden drops with an **immediate tourniquet** (big Low blur step).

5. Produces visible sequential blurring and sharpening consistent with VRAM pressure.
