Day 4 – VRAM Governor (Full Automation)

VramGovernor Project

August 15, 2025

Contents

1 Overview

Day 4 builds upon Day 3 by introducing **full automation** for texel density adjustment. The program now:

- Monitors **free VRAM** and keeps it within a hysteresis band.
- Monitors **average texel density** to maintain visual stability.
- Dynamically adjusts the `GL_TEXTURE_LOD_BIAS` parameter.

The goal: keep rendering smooth without manual control by reducing texture resolution (biasing towards lower mip levels) when memory is low, and restoring sharpness when resources allow.

—

2 New Additions in Day 4

Below are the key new elements added compared to Day 3.

2.1 1. VRAM Monitoring Variables

```
// --- VRAM Governor targets ---
float targetFreeMB = 1024.0f; // desired free VRAM
float bandMB       = 128.0f;  // hysteresis band to
   avoid flapping
```

Why: These define the target free VRAM (e.g., 1 GB) and a **band** of ±128 MB where no change is made. This prevents constant oscillations in texture bias when free VRAM fluctuates slightly.

—

2.2 2. Density Keeper Variables

```
// --- Density keeper ---
float targetDensity = 0.35f; // target normalized
   density
float bandDensity   = 0.03f; // deadband for stability
```

Why: Even if VRAM is fine, texel density can drift (due to scene changes). This keeper makes small corrections to bias for visual stability.

—

2.3 3. VRAM Query Function

```
GLint freeKB = 0, totalKB = 0;
glGetIntegerv(GL_GPU_MEM_INFO_CURRENT_AVAILABLE_MEM_NVX,
    &freeKB);
glGetIntegerv(GL_GPU_MEM_INFO_TOTAL_AVAILABLE_MEM_NVX, &
   totalKB);
float freeMB = freeKB / 1024.0f;
```

Why: Uses the NVIDIA `GL_NVX_gpu_memory_info` extension to read free and total VRAM in KB, converting to MB.

—

2.4 4. Decision Logic (Governor)

```
// VRAM headroom control
if (freeMB < (targetFreeMB - bandMB)) {
    float error = (targetFreeMB - bandMB) - freeMB;
    lodBias += std::min(error * 0.01f, 0.05f); //
       increase blur
}
else if (freeMB > (targetFreeMB + bandMB)) {
    float error = freeMB - (targetFreeMB + bandMB);
    lodBias -= std::min(error * 0.01f, 0.05f); //
       sharpen
}

// Density keeper
if (fabs(avgDensity - targetDensity) > bandDensity) {
    float dErr = targetDensity - avgDensity;
    lodBias -= std::clamp(dErr * 0.05f, -0.02f, 0.02f);
}

lodBias = std::clamp(lodBias, -0.25f, 3.0f);
glSamplerParameterf(samp, GL_TEXTURE_LOD_BIAS, lodBias);
```

Line-by-line:

- If VRAM $<$ target$-$band $\Rightarrow$ increase `lodBias` (more blur).
- If VRAM $>$ target$+$band $\Rightarrow$ decrease `lodBias` (sharpen).
- Step size proportional to error but clamped to avoid sudden jumps.

- Density keeper always runs: corrects average texel density towards target.
- Final bias clamped to $[-0.25, 3.0]$ to avoid extreme settings.

—

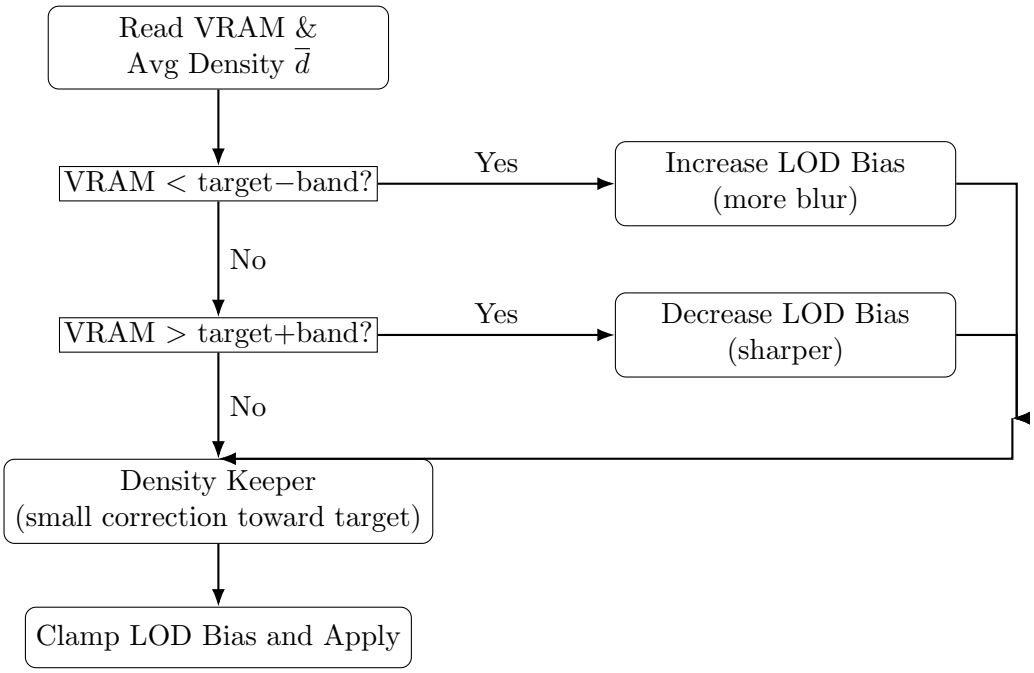
2.5 5. VRAM Load Generation

```
// Create many large textures to consume VRAM
for (int i = 0; i < 50; i++) {
    GLuint tex;
    glGenTextures(1, &tex);
    glBindTexture(GL_TEXTURE_2D, tex);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA8,
                 4096, 4096, 0,
                 GL_RGBA, GL_UNSIGNED_BYTE, nullptr);
    textures.push_back(tex);
}
```

Why: Creates many 4096×4096 RGBA textures without pixel data. OpenGL still allocates VRAM for them, letting us simulate heavy GPU load to trigger the governor.

—

3 Governor Flow Diagram



Governor flow with right-side actions. Branches rejoin at a distant merge to avoid overlaps or crossings.

—

4 Why Blur Saves Performance

- Increasing `LOD_BIAS` forces the GPU to sample lower mip levels.
- Lower mips = larger texels ⇒ fewer fetches per pixel.
- This reduces:
  - Memory bandwidth usage.
  - Cache misses.
  - VRAM consumption.
- When VRAM headroom is low, this recovers performance.

—