How Texel Density Is Lowered with Mipmaps and LOD Bias

Rameel Malik

August 12, 2025

4-Step Explainer

1. **Before rendering: build the mipmap pyramid.**
   After uploading the base texture (e.g., 4096×4096), calling `glGenerateMipmap(GL_TEXTURE_2D)` creates a chain of downscaled images stored in VRAM:

   $$\text{Level 0: } 4096^2,\ \text{Level 1: } 2048^2,\ \text{Level 2: } 1024^2,\ \text{Level 3: } 512^2,\ \ldots,\ \text{Level } n : 1^2.$$

   These are precomputed once and reused every frame.

2. **During rendering: the GPU chooses a mip level automatically.**
   For each fragment (pixel), the texture sampler estimates the on-screen footprint from UV derivatives and picks a mip level often summarized by

   $$\lambda \approx \log_2(\rho),$$

   where ρ encodes how fast the UVs change (times texture size). Intuition: large on-screen ⇒ small λ (higher-res level); small on-screen ⇒ large λ (lower-res level).

3. **Why it looks blurrier at distance (and why that's good).**
   Lower mip levels are prefiltered averages of larger texel neighborhoods (Level 1 ≈ 2×2, Level 2 ≈ 4×4, etc.). Sampling them reduces shimmering/aliasing and lowers memory bandwidth, because fewer/smaller texels are fetched per screen pixel.

4. **LOD bias: nudging the chosen level for performance or sharpness.**
   We apply a user-controlled offset to the hardware's choice:

   $$\lambda_{\text{effective}} = \lambda_{\text{hardware}} + \text{bias}.$$

In code: `glSamplerParameterf(samp, GL_TEXTURE_LOD_BIAS, bias)`. A **positive** bias (e.g., $+1.0$) picks lower-resolution mips sooner ($\approx$ one level lower, half-res each dimension) $\Rightarrow$ *lower texel density, higher performance.* A small **negative** bias (e.g., -0.25) keeps higher-res mips a bit longer $\Rightarrow$ *sharper but costlier.*