MIP-Mapping With Procedural and Texture-Based Magnification

Markus Hadwiger* Thomas Theußl† Helwig Hauser* Meister Eduard Gröller†

*VRVis Research Center
Vienna, Austria
{Hadwiger|Hauser}@VRVis.at

†Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria
{theussl|meister}@cg.tuwien.ac.at

1 Introduction

The flexible programmability of current consumer graphics hardware at the pixel level allows to use higher-order, e.g., cubic, magnification for texture filtering as an alternative to the hardware-native linear interpolation. Higher-order filter kernels can either be evaluated procedurally [NVIDIA 2002] or be stored in texture maps [Hadwiger et al. 2002].

When MIP-mapping is used for minification, the higher-order magnification filter degenerates to nearest-neighbor interpolation for lower-resolution MIP-map levels (figure 1). We show how to remove the resulting artifacts (figure 2) by automatically adapting the magnification filter to the actual MIP-map level used for a given pixel.

2 Algorithm

Both procedural and texture-based higher-order magnification filters require exact matching of input texture samples to filter weights. In order to achieve this, the exact resolution of the input texture has to be known, which can actually be different from pixel to pixel in the presence of MIP-mapping. On current hardware, it is not directly possible to determine the MIP-map level used for a given pixel, and thus the actual per-pixel input texture resolution. For interpolation between adjacent MIP-map levels, even two input texture resolutions have to be considered for a single pixel. In order to get access to all the required MIP-map level information in the pixel shader, we use an additional four-channel *meta MIP-map* that contains information about the MIP-map itself:

- U and V texture coordinate scaling factors for matching filter width and MIP-map level, specified relatively to the base level to stay within a $[0, 1]$ range (e.g., 0.25 for a MIP-map level of size 16 and a base level of size 64).
- U and V texture coordinate offsets to get from one texel to the next (e.g., 1/64 for a MIP-map level of size 64).

The meta MIP-map is used in the pixel shader for adjusting the interpolated texture coordinates to reflect the per-pixel input texture resolution. This allows to implement a `GL_CUBIC_MIPMAP_NEAREST` minification filter. The same scheme can also be applied for interpolation between MIP-map levels, thus yielding a `GL_CUBIC_MIPMAP_LINEAR` filter.

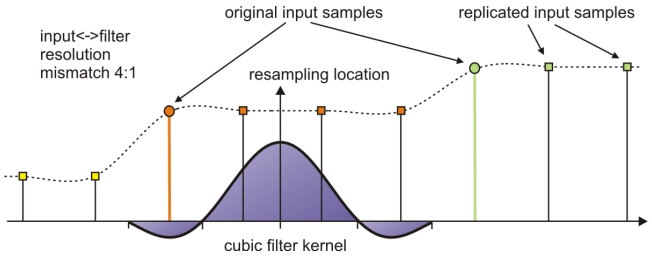

Figure 1: The input samples used by the reconstruction filter depend on a proper scaling of the filter kernel with respect to the input texture, which can only be done if the input texture resolution is known. When MIP-mapping is used, the filter kernel is usually scaled for the base level, leading to an incorrect scaling for lower-resolution levels. For these levels, this leads to replication of input samples with respect to the reconstruction filter. The filter effectively uses fewer and fewer input samples, more and more degenerating to nearest-neighbor interpolation. For a cubic filter, degeneration to nearest-neighbor already starts two MIP-map levels away from the base level, i.e., for a resolution mismatch of 4:1, which is depicted here.

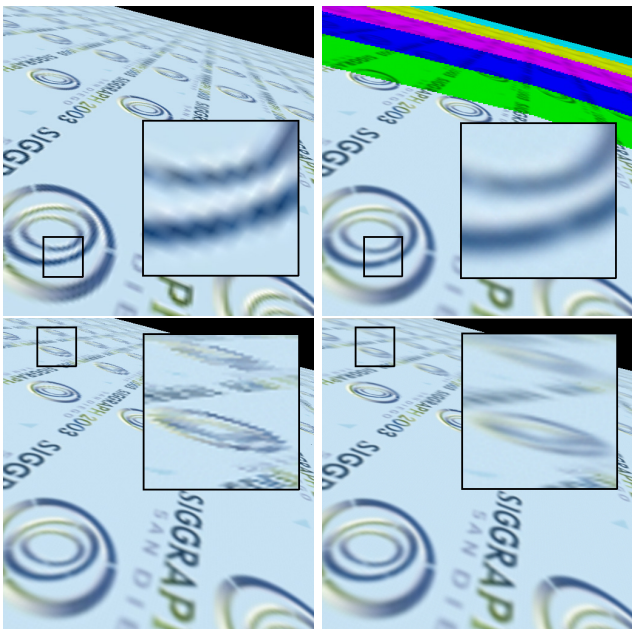

Figure 2: Linear interpolation and MIP-mapping (top left). Cubic reconstruction improves magnification (top right; MIP-map levels color coded), but degenerates to nearest-neighbor interpolation at lower-resolution MIP-map levels (bottom left). Cubic magnification with correct MIP-map filtering at all levels (bottom right).

References

Hadwiger, M., Viola, I., Theussl, T., and Hauser, H. 2002. Fast and flexible high-quality texture filtering with tiled high-resolution filters. In *Proceedings of Vision, Modeling, and Visualization 2002*, *VMV'02*, 155–162.

NVIDIA, 2002. Cg effects browser 5.0, bicubic texture filtering example. See `http://developer.nvidia.com/`.