Jörn Loviscach

Efficient Magnification of Bi-Level Textures
Objective

- Bi-level (aka binary) images such as road signs need sharp texturing
- Do that fast; consume only little memory

Hi-res | Lo-res | DXT1 | Presented method
---|---|---|---
2 x mem | same mem as lo-res
Outline

• Related work
• Thresholding and how to make it work
• Antialiasing on magnification
• MIP mapping on minification
• Optimizing the lo-res texture
• Results
• Future Work
Related Work

- Textures with hard edges:
  - Ramanarayanan, Bala, Walter: Feature-based textures, 2004
  - Tumblin & Choudhury: Bixels, 2004

- ... on the GPU:
  - Sen: Silhouette maps, 2004
Thresholding and How to Make It Work (1)

- Address only bi-level textures
- Use thresholding in the pixel shader for a bilinearly interpolated low-resolution texture
Thresholding and How to Make It Work (2)

- Task 1: Antialiasing on magnification
- Task 2: MIP-Mapping on minification
- Task 3: Optimize the low-res texture
Antialiasing on Magnification

- Soft threshold with transition region

- Fast and cheap:
  - \( c = \text{clamp}(0.5 + a^*(t-0.5)) \)
  - \( ddx, ddy \) to determine size

\( \text{inverse size of pixel preimage} \)
MIP-Mapping on Minification (1)

- On minification, sharp edges have to be increasingly blurred
- Use regular MIP mapping; all MIP levels as usual except the finest one
MIP-Mapping on Minification (2)

- Don’t want grey halos on transition between MIP levels 0 and 1: Prohibit medium gray values (say, 112 to 143) in level 0
Optimizing the Lo-Res Texture (1)

- Input: high-resolution bitmap
- Output: low-resolution texture
Optimizing the Lo-Res Texture (2)

- Extract intersections/angles at edges of lo-res texels
- Try to mimic intersections/angles with thresholded lo-res texture
- Overdetermined system
- Seek optimal solution
Optimizing the Lo-Res Texture (3)

- Authoring software
  - Optimization in background thread
  - Edit extracted intersections/angles
Optimizing the Lo-Res Texture (4)

- Which metric to use for optimization?
  - Favor accuracy of position: bumpy diagonals
  - Favor accuracy of angles: varying stem widths
  - Hence: Favor positions only at nearly horizontal or vertical intersections
Results

- Sharp edges in spite of small textures
- Speed: 800 MP/s on GF 6800 GT (14 instructions incl. one texture request)
- Visually robust typefaces reproduced well, but there are obvious limits
Future Work

- “Intelligent” metric for optimization?
- Reverse engineer multi-color signs

\[
\text{to generate sharp intersections}
\]
Questions?