This paper introduces Metatexturing: a semi-procedural surface material system that leverages the power of both texture-image-based and procedural material systems.

Like a purely procedural system, Metatexturing can produce many variations of a surface archetype without the need for new texture images for each variation. This saves memory. Like a traditional pipeline, the parameters that drive the Metatexturing method are entirely defined by a few texture images and simple shader constants. This saves artist time and leverages existing skills.

1 Introduction and Background

The real-time computation of Perlin Noise is beyond current GPU power, but an off-line created noise texture can be used to drive the computation of per-pixel surface material properties, such as color, surface normal, and specular highlights [PER04].

If, instead of a noise texture, the driving texture is a set of two-dimensional texture maps, the same methods can be used to compute surface material properties. This differs from a traditional pipeline as the values in the texture images do not directly define the surface properties, but serve as parameters for functions that compute them.

In the pipeline outlined below, three textures are used. The structure metatexture encapsulates information about the large-scale structure of the material. The detail metatexture denotes the fine-scale detail of the material. The color helper metatexture, a small one-dimensional image, functions as a color lookup table.

![Figure 1. A block diagram of the metatexturing pipeline. The “Offset Lookup” block at the bottom center represents additional modulated lookups into the structure and detail textures using different texture coordinates as noted.](image2)

The second step is to perform a lookup on the detail metatexture, resulting in the detail value, also with up to four variations packed into an RGBA image. Here, the structure value is used instead of one of the shader constants to choose the desired variation. The reason for this is to allow the detail to be dependent on the overall structure, such as brick versus mortar.

The surface value is produced by a biased addition of the detail value to the structure value. The surface value is then used as the texture coordinate for a lookup into the color helper metatexture, and the resulting color is used as the base color of the fragment.

To compute the surface normal, the surface value is treated as a height value, and additional, offset lookups into the structure and detail maps produce new height values, and the deltas in the x or y direction become the normal perturbation for that axis. The result is a tangent-space normal that can then be used in standard lighting calculations.

3 Creating Variations

Per-object variation can be achieved by varying the three shader constants that control the LERP between the image channels of the structure and detail metatextures. Significant appearance differences can be achieved with this technique alone.

The Metatexture images can be individually swapped out to allow for more variation. Replacing The 1D color helper metatexture affects color across the entire surface. This is useful for materials such as ceramic tile floors, which differ greatly in color, but share other surface parameters. Changing structure metatextures allows more structure variations, but keeps the fine details, which is useful for objects that need to fit seamlessly together. Swapping detail metatextures creates more detail variations, allowing changes such as clean and new bricks versus chipped and old bricks.

![Figure 2. Tile floor variations achieved through replacing the color helper texture and/or changing the structure and detail shader constants.](image3)

4 Conclusions and Future Work

The current Metatexture implementation shows promise as an alternative to both traditional texture-driven and procedural pipelines. Future work will extend the method to incorporate the calculation of more surface parameters, and to expand the variety of surface archetypes that can be modeled.

References