Implementing Vector-Based Texturing In RenderMan

Contact
John Haddon
NCCA
Bournemouth University
United Kingdom
theboyhaddon@hotmail.com

Ian Stephenson
NCCA
Bournemouth University
United Kingdom

The predominant method of texturing for production is using bitmap image files in conjunction with procedural shading. However, the storage demands of bitmap images increase greatly with resolution, and procedural techniques are typically unsuitable for generation and fine control of complex figures.

Vector graphics provide resolution-independent, scaleable images, typically with low file sizes, and are easily designed using available software. This makes them ideal for use in some texturing situations, particularly where it is necessary to incorporate imagery in a graphic style. This sketch presents the implementation of vector-based texturing in a RenderMan renderer.

API

As implemented, vector graphics lookups appear to the shader writer much as the built-in texture() calls do. A family of new shadeops of the general form vtexture (uniform string filename, float u, float v, string filtertype) return texture color and alpha information. Beyond this, the shader writer requires no knowledge of the system's internals.

Implementation

vtexture() is implemented as two DSO shadeop calls. The first, called once per grid with uniform parameters, ensures that the required texture is loaded into a texture cache and marked as being current. At this point, the texture is in its idealised, resolution-free form.

A second shadeop call, executed once per micro-polygon, receives areas to be filtered as arbitrary quads in texture space and returns texture color and alpha information. This is achieved by generating and maintaining a cache of tiles (rasterised sections of texture) and filtering them appropriately. Tiles are rasterised at resolutions adapted to the lookups requested, and a new tile typically includes a reasonable area surrounding the current lookup area. This means that there is a fair chance of a tile that is suitable for the following lookups being already present in the cache. Rasterisation is a computationally significant process, so the effectiveness of this caching is essential to performance.

Example Usage

vtexture() was employed in rendering a sequence that shows a track into a globe, starting at a point where the whole earth is visible and ending on a small high-resolution section, specifically the Isle of Wight. For comparison, the same sequence was texured using an 8,000 x 8,000-pixel bitmap.

Both sequences were net-rendered with PRMan 3.9. The vector version required approximately three times the computing time of the bitmap. The vector texture was approximately 800K in size, whereas the bitmap was significantly larger (almost 250 MB of uncompressed data). However, most significantly, the vector texture provides resolution several orders of magnitude greater than that achievable with a bitmap of this size (Figure 1).

Considerations

vtexture() has been implemented and tested with PRMan. The Reyes algorithm typically generates successive texture lookups that are adjacent in texture space, as they are generated from adjacent grid points. Other rendering algorithms, particularly ray tracing, are unlikely to generate such adjacent lookups, resulting in much less effective caching of tiled data. Presumably, this would significantly limit performance.

Currently, rendering with the system is significantly slower than with substitute bitmaps. Although this is likely to always be the case, it is believed that optimisations, particularly at the rasterising and filtering stages, could significantly increase performance.

Vector texturing is by no means a panacea. Textures of a photographic nature are simply not representable in a vector form. Text and graphic shapes are among those most suitable for vector description, and these could be augmented with procedural techniques in situations that demand greater photo-realism.

Conclusion

It has been shown that vector-based texturing can be successfully implemented under RenderMan as an extension to the shading language. The system described operates with viable performance and over a significant range of resolution. This demonstrates the potential value of vector textures in production.

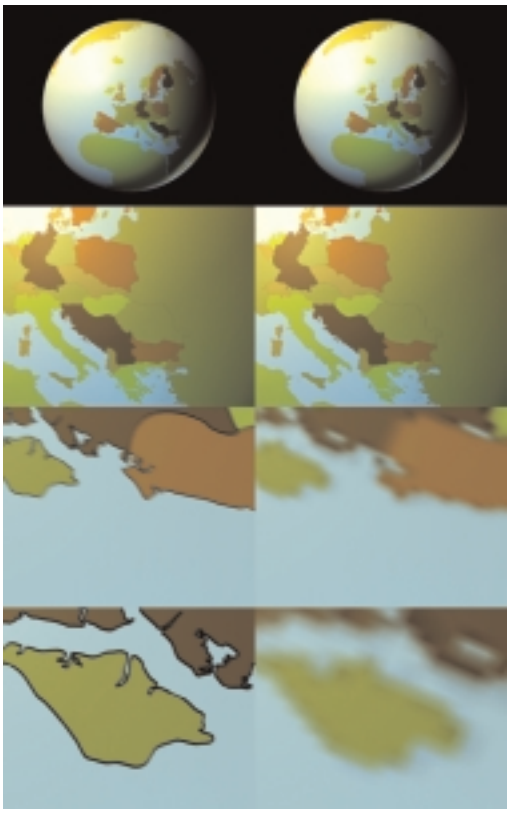

Figure 1: The vector texture (left) provides significantly increased resolution in comparison with the test bitmap texture (right).