Polymorphic Pixels

Robin Dunlop David Callele

Department of Computer Science
University of Saskatchewan

Abstract

We present an alternative to traditional texture compression techniques whereby a clear data channel is embedded within a texture using steganographic techniques. The resulting pixels are polymorphic – they can be used without decompression and the embedded data can be extracted and used for alternative purposes. The technique has fewer degenerate cases than DXT, does not suffer from high-frequency artifacts, but has lower compression ratios.

Key words: Steganography, compression, polymorphism

1 Introduction

Media assets such as textures require significant storage space and communications channel bandwidth. Compression techniques can reduce these requirements at the cost of runtime data decompression.

Based on earlier work by Callele [1], we present an alternative to traditional compression that uses steganographic techniques to embed an alternate data stream within a texture. The texture data is then used for multiple purposes at runtime – each pixel is polymorphic.

2 Background

Image compression techniques such as DXT [2] assume that the original image exhibits sufficiently strong spatial locality that the image can be compressed on a local basis. DXT achieves a 6:1 compression ratio on a 4 by 4 texel region of a 24-bit RGB image by representing the maximum and minimum values in the texel region by 16-bit (5-6-5) approximations and quantizing each region member to one of 4 levels. When the image is decompressed, it is effectively converted to a 16-bit (5-6-5) approximation to the original image. Concatenating the 5-6-5 result with the output of a noise function to achieve an 8-8-8 result approximates natural noise and reduces banding artifacts. If the original image does not exhibit strong spatial locality then significant artifacts can be introduced. While these high-frequency artifacts can sometimes be reduced by adding noise to the original image *before* compression, the tradeoff between noise and fidelity may not be acceptable and the image may not compress satisfactorily using this technique.

Reconstituting the least significant bits of the original image with the output of a noise function has a parallel in steganography [3]. One form of steganography hides information (such as a digital watermark) within an image by replacing some number of the least significant bits in the original image with the information to be hidden. The hidden information is embedded in the original image as if it were a noise function.

Both DXT decompression and steganography add noise to the image. The viewer usually perceives this noise as adding complexity to the image. In the absence of *a priori* knowledge, a reference image, or large–scale artifacts (*e.g.* banding), it is difficult for a viewer to identify the noise as errors.

3 Polymorphic Pixels

We illustrate techniques for converting a texture to polymorphic pixels by way of two examples. In both examples, error minimization is performed by comparing the original image to the modified image on a per-texel basis.

For each color channel in a texel, let M be the most significant bits of the original image, L the least significant bits of the original image, E the bits to be embedded by replacing L, and M' the most significant bits of the final image. Let m be the length (in bits) of M and M', l be the length of L, and e the length of E. Let $\circ$ be the concatenation operator such that $M \circ L$ represents the concatenation of the bits in the original image.

The algorithm for embedding an arbitrary bit pattern in a channel of an RGB image can be stated as follows.

$$
\begin{aligned}
&M' = M \\
&If((M \circ L) - (M \circ E)) > 2^{e-1} \\
&\quad M' = M + 1 \\
&If((M \circ L) - (M \circ E)) < -2^{e-1} \\
&\quad M' = M - 1 \\
&FinalChannelValue = M' \circ E
\end{aligned}
$$

The mapping used in Figure 1(e), between the bits to be embedded b and the texel bits in each channel, is as follows:

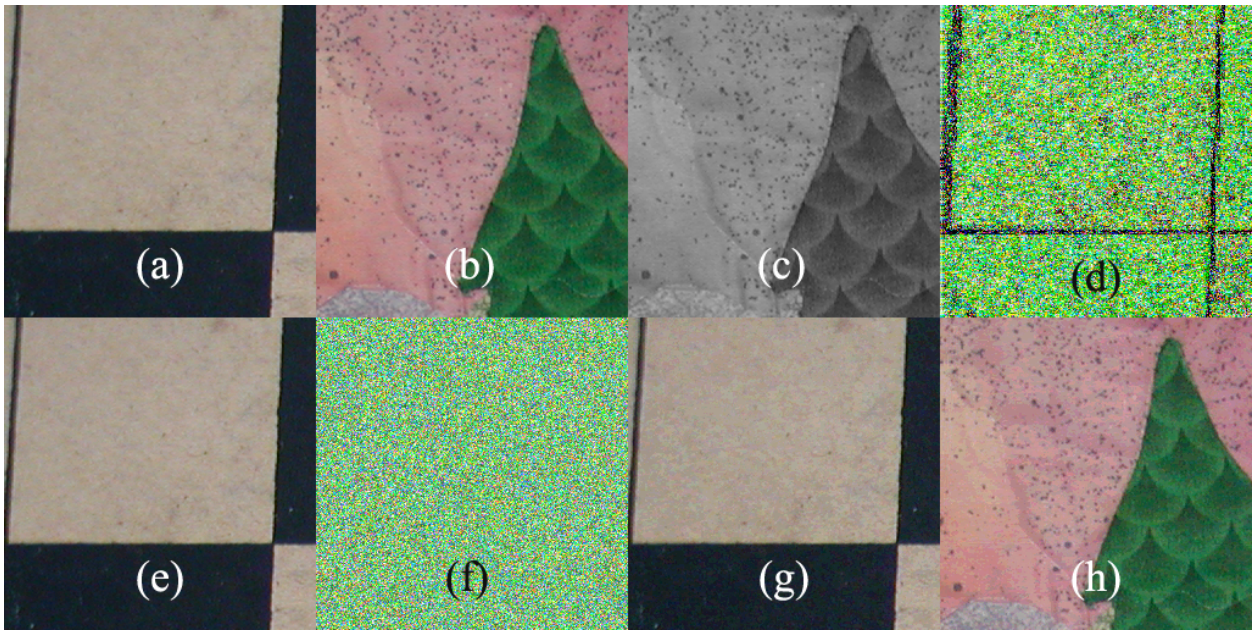

Figure 1: Sample Results

$$\begin{array}{cccccccc} b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\ R_2 & R_1 & R_0 & G_1 & G_0 & B_2 & B_1 & B_0 \end{array}$$

Alternatively, two images A and B can be embedded in each other (*mutually embedded*) by creating a 4-4-4 representation of each image then compositing the result in either format as shown. Error minimization is applied and the image in the least significant bits can be extracted by mirroring the bits in each channel.

$$\begin{array}{cccccccc} B_7 & B_6 & B_5 & B_4 & A_4 & A_5 & A_6 & A_7 \\ A_7 & A_6 & A_5 & A_4 & B_4 & B_5 & B_6 & B_7 \end{array}$$

4 Results

Figure 1 illustrates sample results. Subimages (a) and (b) are 24-bit master images, (c) is an 8-bit version of (b). Subimage (d) is the error after compressing and decompressing (a) using DXT; the image has been post-processed to enhance the results for print. Subimage (e) is the result of embedding (c) in (a) using a 3-2-3 embedding; (f) is the post-processed error in (e) compared to (a). Subimages (g) and (h) are the result of mutually embedding (a) and (b). Qualitatively, all images appear satisfactory.

5 Conclusions and Future Work

The benefits of this technique include greater image fidelity than DXT with a computationally simpler algorithm. The images can be used as they are at runtime, there is no need to perform a decompression phase (the embedded data can be extracted with bitmasks and concatenation operations). There are fewer degenerate cases and the technique does not suffer from high-frequency artifacts. The technique can be implemented on current generation GPUs via texture lookup tables and on next-generation GPUs directly in code. Though the examples shown illustrate embedding one texture in another, the embedded data can be used for any purpose.

The drawbacks of this technique include lower compression than DXT, reducing storage and bandwidth savings. Managing the combined data adds another level of complexity to the rendering pipeline.

Acknowledgements

Our thanks to nVIDIA for their support of our research.

References

[1] David Callele. Unpublished manuscript: Real-time Displacement Mapping, 2000.

[2] Microsoft Corporation. *DirectX SDK Documentation*. Microsoft Corporation, 2005.

[3] F. Petitcolas, R.J. Anderson, and M.G. Kuhn. Information hiding: A survey. *Proceedings of the IEEE*, 87(2):1062–1078, July 1999.