Decomposition of 32 bpp into 16 bpp Textures with Alpha

Nobuki Yoda Takeo Igarashi
The University of Tokyo

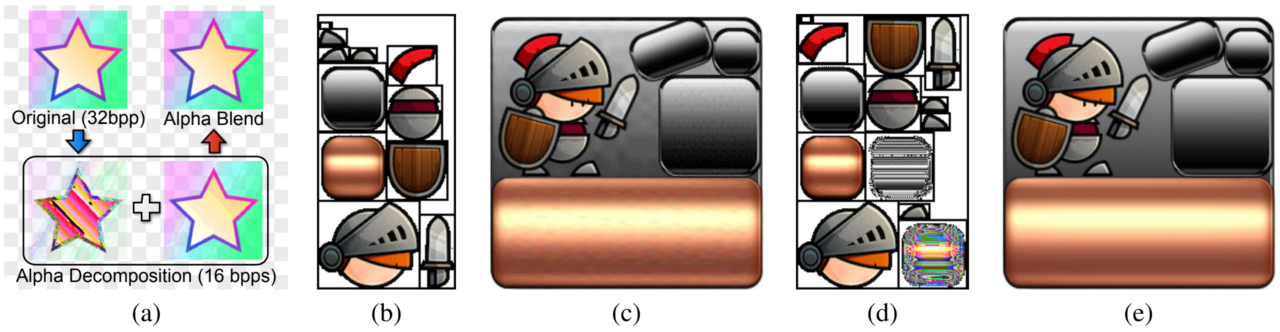

Figure 1: *(a) An overview of our method. (b, c) A normal 16 bpp sprite sheet and its rendering result. It contains noise. (d, e) A 16 bpp sprite sheet with selectively alpha-decomposed textures and its rendering result. The noise is reduced. (b-e) Knight (© Game Pro Market) and metal squares (© Tri-Angulum Studios) are from Unity Asset Store (https://www.assetstore.unity3d.com/).*

1 Introduction

In 2D game graphics, textures are packed into a single texture called a sprite sheet in order to achieve efficient rendering. The sprite sheet can be compressed to save memory by using various compression methods such as block-based compressions and 16 bpp (bits per pixel) tone reduction. These methods are not without some problems, though. Block-based compressions are GPU-dependent, and high-quality compressions such as ASTC [Nystad et al. 2012] are often unavailable on mobile devices. 16 bpp tone reduction–often used with dithering–can create undesirable noise when it is scaled up (Figure 1c).

In this work we propose a new method called alpha decomposition for achieving better trade-off between texture quality and memory usage in 2D sprites. The artist first decomposes a 32 bpp texture into two 16 bpp textures with alpha and then packs them into a single sprite sheet together with normal 16 bpp textures. When rendering, the two decomposed 16 bpp textures are simply layered and alpha-blended, reproducing the source 32 bpp as faithfully as possible (Figure 1a). This method does not require customized rendering pipelines, and thanks to sprite batching is able to achieve efficient rendering, as opposed to a naive method that use separate sprite sheets for 32 bpp and 16 bpp textures.

2 Alpha Decomposition

First, we define a novel error metric for RGBA color called integrated square error over all possible background colors (ISEABC):

$$ISEABC\,(a,b) = \int_{x\in[0,1]^3} \left|\boldsymbol{C}\,(a\odot x) - \boldsymbol{C}\,(b\odot x)\right|^2 dx$$

where a and b are the compared colors with alpha, x is an opaque background color , the operator $\odot$ is alpha blending [Wallace 1981], and $\boldsymbol{C}\,(c)$ is the RGB vector of c. Since ISEABC is an extension of square error of RGB vector which is used for MSE and PSNR, these metrics can be extended for RGBA color space by ISEABC.

Next we define the alpha decomposition using the error metric. Let t be a source 32 bpp texture, f and b be the decomposed 16 bpp textures, and $\boldsymbol{u}$ be a UV texture coordinate. The decomposition is then formulated as the following minimization problem:

$$\underset{\langle f,b\rangle}{\arg\min} \int_{\boldsymbol{u}\in[0,1]^2} ISEABC(\mathcal{F}\,(t,\boldsymbol{u})\,, \mathcal{F}\,(f,\boldsymbol{u}) \odot \mathcal{F}\,(b,\boldsymbol{u}))d\boldsymbol{u}$$

where $\mathcal{F}$ represents texture filtering and wrapping. We propose approximation algorithms for the problem. If $\mathcal{F}$ is a nearest-neighbor filter, the problem can be solved independently for each pixel. If it is a bilinear filter, the filtering results depend on neighboring pixels, so we minimize the local errors sequentially from the left-bottom pixel to the top-right pixel, referring to previous pixels. If alpha is fixed, ISEABC can be minimized independently for each RGB component. We first choose some candidates for the resulting alpha pair from all possible pairs by thresholding the alpha error, and then search for the best RGB pair for each candidate pair.

Our experimental results are summarized as follows. Computation time for decomposition was approximated 3 minutes for Figure 1a (128^2 pixels). We measured image quality when applied to *Kodak* images[1] and the average of our method (PSNR = 53 dB) was better than that of 8 bpp ASTC (PSNR = 45 dB). We also measured rendering speed for a highly dense scene and our method (24 fps) was equivalent to normal 16 bpp (24 fps) and faster than a naive method of using separate sprite sheets for 16 bpp and 32 bpp (8 fps).

References

NYSTAD, J., LASSEN, A., POMIANOWSKI, A., ELLIS, S., AND OLSON, T. 2012. Adaptive scalable texture compression. In *Proc. EGGH-HPG'12*, 105–114.

WALLACE, B. 1981. Merging and transformation of raster images for cartoon animation. In *Proc. SIGGRAPH '81*, 253–262.

[1]Kodak lossless true color image suite (http://r0k.us/graphics/kodak/)

SIGGRAPH 2015 Posters, August 09 – 13, 2015, Los Angeles, CA.
ACM 978-1-4503-3632-1/15/08.
http://dx.doi.org/10.1145/2787626.2792610