RGBdiv8 texture/framebuffer HDR compression

Method

This is a simple, yet effective way of storing medium/high dynamic-range data in low dynamic-range resource by using the alpha-channel to extend the range of the RGB-channels.

Encoding:

- Find the largest scalar value among the RGB-channels and 1.0
- Divide the RGB-value (with 1.0 in the alpha channel) with it

Decoding:

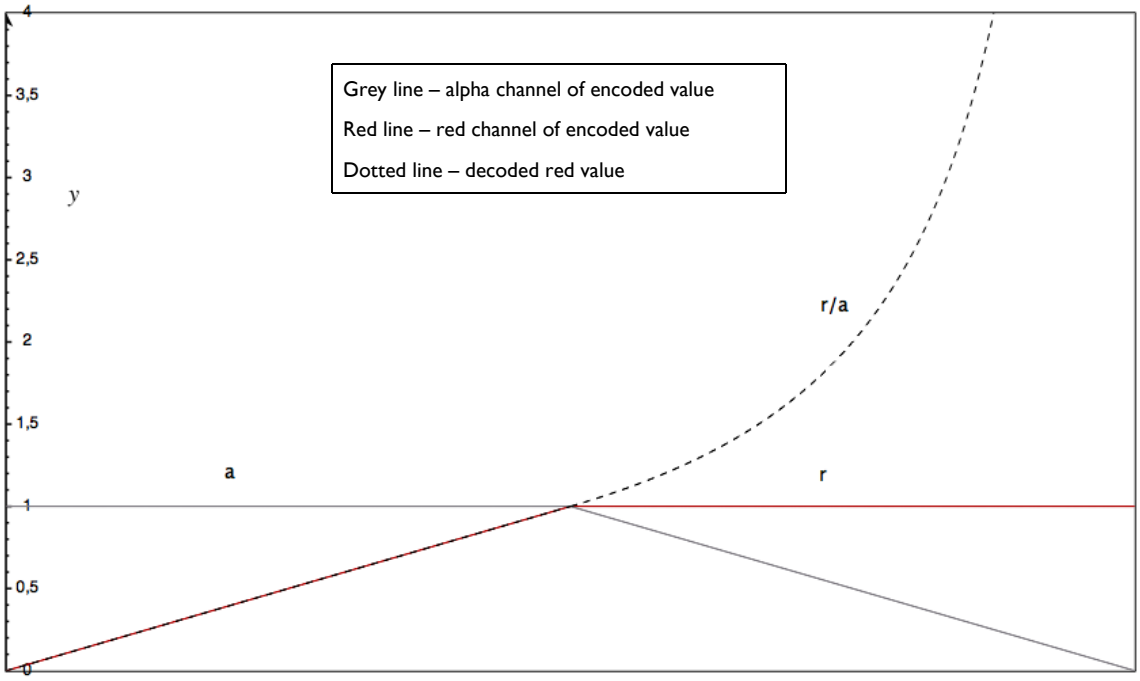
- Divide the RGB-channels with the alpha channel. (only 2 instructions on ps2.0)

HLSL implementation

```
float4 encode_RGBdiv8(float4 col)
{
    float max_RGB1 = max(max(1.f, col.r),max(col.g,col.b));
    return float4(col.rgb,1) / max_RGB1;
}

float4 decode_RGBdiv8(float4 col)
{
    return col / col.a;
}
```

Graph



By studying the encoded representation of a single channel it's easy to see that while the precision in the [0..1] range is as normal, the division by alpha-channel provides an extended range for larger numbers.

Benefits

- Precision in the [0..1] range is identical to RGB8
- The extended precision is right where we usually want it, dense close to 1 and sparser as it gets higher up
- Quantitization of the alpha channel will only affect luminance. Hue and saturation is unaffected.
- No discontinuities caused by reliance on simultaneous switching of RGB & A (unlike f.e. RGBE)
- Works reasonably well with blending/interpolation. (as long as its lerp)
- Both encoding & decoding is relatively cheap and works well with ps2.0-level hardware

Possible issues

- Additive blending is contradictary on the alpha-channel, need workaraounds
- Still sensitive to precision, scale responsibly

Other thoughts

When doing blending, keep in mind that it's possible (in Direct3D) to specify different blend functions for the RGB- and A-channels. This lets you work around most blending limitations with some sacrifices and ingenuity.

The precision in the low end is probably not enough to use linear framebuffers without noticable quantitizing (just like normal RGB8). However one could get around this issue by scaling so that a larger amount of the possible representations are in the [0..1] range. For example, by letting the [0..1] range represent [0..0.25] this range will get equivalent to 10-bits of precision at the expense of the precision in the remaining dynamic range.