adding alpha values to bilinear resizing algorithm

Asked 16 years ago Modified 16 years ago Viewed 2k times

3

So I'm trying to take a bilinear interpolation algorithm for resizing images and add in alpha values as well. I'm using Actionscript 3 to do this, but I don't really think the language is relevant.

The code I have below actually works really well, but edges around "erased" regions seem to get darker. Is there an easy way for it to not include what I can only assume is black (0x00000000) when it's finding its average?

Code:

```
x_ratio = theX - x;
y_ratio = theY - y;
x_opposite = 1 - x_ratio;
y_opposite = 1 - y_ratio;

a = getPixel32(x, y);
be =getPixel32(x + 1, y);
c = getPixel32(x, y + 1);
d = getPixel32(x + 1, y + 1);
alph    = (t(a)  * x_opposite  + t(be)   * x_ratio) * y_opposite + (t(c) *
x_opposite  + t(d) * x_ratio) * y_ratio;
red     = (r(a)  * x_opposite  + r(be)   * x_ratio) * y_opposite + (r(c) *
x_opposite  + r(d) * x_ratio) * y_ratio;
green   = (g(a)  * x_opposite  + g(be)   * x_ratio) * y_opposite + (g(c) *
x_opposite  + g(d) * x_ratio) * y_ratio;
blue    = (b(a)  * x_opposite  + b(be)   * x_ratio) * y_opposite + (b(c) *
x_opposite  + b(d) * x_ratio) * y_ratio;
```

Image of the effect: http://beta.shinyhammer.com/images/site/eraser_pixelborders.jpg

Posting code of solution!

```
a = getPixel32(x, y);
be =getPixel32(x + 1, y);
c = getPixel32(x, y + 1);
d = getPixel32(x + 1, y + 1);
asum = (t(a) + t(be) + t(c) + t(d)) / 4;

alph    = (t(a)  * x_opposite  + t(be)   * x_ratio) * y_opposite + (t(c) *
x_opposite  + t(d) * x_ratio) * y_ratio;
red     = ((r(a) * t(a) * x_opposite  + r(be) * t(be) * x_ratio) * y_opposite +
(r(c) * t(c) * x_opposite  + r(d) * t(d) * x_ratio) * y_ratio);
red = (asum > 0) ? red / asum : 0;
green   = ((g(a) * t(a) * x_opposite  + g(be) * t(be) * x_ratio) * y_opposite +
(g(c) * t(c) * x_opposite  + g(d) * t(d) * x_ratio) * y_ratio);
green = (asum > 0) ? green /  asum : 0;
blue    = ((b(a) * t(a) * x_opposite  + b(be) * t(be) * x_ratio) * y_opposite +
```

```
(b(c) * t(c) * x_opposite  + b(d) * t(d) * x_ratio) * y_ratio);
blue = (asum > 0) ? blue / asum : 0;
```

flash actionscript-3 image resize interpolation

Share
Improve this question
Follow

edited Dec 7, 2008 at 18:55

asked Dec 7, 2008 at 5:51

Brent
23.7k ● 10 ● 47 ● 49

2 Answers

Sorted by: Highest score (default)

7

You need to multiply each of your r,g,b values by the corresponding alpha before working with them, then divide the values by the final alpha when you're done. It's easy to imagine the effect this will have when one of the pixels has an alpha of zero - each of the r,g,b values will be multiplied by zero, so it won't matter what their original values are at all! They won't contribute to the final result.

You'll need to add a check so that you never divide by zero, and make sure that roundoff error doesn't push your final values past the upper limit.

Often images will be stored in memory with the alpha already multiplied into the r,g,b values - this is called premultiplied alpha (Wikipedia reference).

Share Improve this answer Follow

answered Dec 7, 2008 at 8:14

Mark Ransom
308k ● 44 ● 416 ● 647

4

This is one of those areas where the use of pre-multiplied alpha is a huge advantage. With pre-multiplied alpha, you can think of the RGB components of your image as having been pre-composited on black in advance.

This removes many of the troublesome effects associated with any kind of image processing on images with alpha, as well as giving a faster compositing algorithm.

For "non-pre-multiplied alpha", the familiar LIRP (linear interpolation) compositing algorithm is

```
d = Kf + (1-K)b
```

... where K is the alpha of the foreground, f is the foreground value and b is the background.

For "pre-multiplied alpha", you use

```
d = f + (1-K)b
```

... which removes one of the multiplies, which can make the process faster (and cheaper in hardware). The key paper to read is Porter and Duff's - "Compositing Digital Images"

If you're using Windows, the Win32 AlphaBlend function requires premultiplied alpha, so it's good to stay in the premultiplied domain whenever possible.

Share Improve this answer Follow

answered Dec 7, 2008 at 10:24

Roddy

67.9k ● 44 ● 170 ● 280