Subject: saving a paletted PNG
Posted by lindquist on Sun, 29 Oct 2006 19:42:07 GMT
View Forum Message <> Reply to Message

Hi folks.
I have a problem with saving a paletted png image. From looking in plugin/png it seems that it is actually handling paletted images, but I always end up with a 24bit png. The problem seems to be PixelArrayToImage which converts my PixelArray (8bpp) to a Image.

I have tried without success to create a Draw that will maintain the paletted format.

Can anyone help me out here, I'm not sure if the support is there or if I need to do some hacking.

Any clues/pointers would be very much appreciated.
I need the exported png to be paletted!

Thanx, Tomas

Subject: Re: saving a paletted PNG
Posted by lindquist on Sun, 29 Oct 2006 19:53:46 GMT
View Forum Message <> Reply to Message

this is basically what I'm doing: (greatly simplified)

```
void writePNG(const String& filename)
{
 Vector<Color> pal;
 pal.Add(Color(0,0,0));

 PixelArray pix(100, 100, 8, 1, NULL, pal);

 for (int i=0; i<10000; ++i)
  pix.pixels[i] = 0;

 Image img = PixelArrayToImage(pix);
 PngEncoder::New()->SaveImageFile(filename, img);
}
```

Subject: Re: saving a paletted PNG
Posted by mirek on Sun, 29 Oct 2006 19:55:20 GMT
View Forum Message <> Reply to Message

PixelArray is now obsolete, sorry. I recommend downloading the dev version of U++, raster Image handling is vastly refactored.

In the new version, PNGEncoder has Bpp method (bits per pixel). Put there 8 to have 256 colors palette. If you have existing palette you want to retain, use SetPalette.

Mirek

Subject: Re: saving a paletted PNG
Posted by lindquist on Sun, 29 Oct 2006 20:02:15 GMT
View Forum Message <> Reply to Message

Thanx for the very quick reply. I will grab the latest dev version. Hope it wont break too much

---

Subject: Re: saving a paletted PNG
Posted by lindquist on Sun, 29 Oct 2006 23:10:58 GMT
View Forum Message <> Reply to Message

I'm having some more problems.
I have this test code:

```
void WritePNG(const String& filename)
{
 ImageBuffer ib(256, 256);
 ib.SetKind(IMAGE_OPAQUE);

 RGBA pal[256];
 for (int i=0; i<256; ++i)
 {
  pal[i] = Color(i,i,i);
 }

 int idx = 0;
 for (int y=0; y<256; ++y)
 {
  RGBA* line = ib[y];
  for (int x=0; x<256; ++x)
  {
   *line = pal[idx%256];
   ++line;
   ++idx;
  }
 }

 PNGEncoder png(8);
 png.SetPalette(pal);
```

```
 png.SaveFile(filename, ib);
}
```

I would assume this code would generate a nice horizontal gradient, but it doesn't. Instead I get something that seems a bit like it's converted to 5bit colors. (every 8 pixels it jumps 16 indexes in the palette)

Subject: Re: saving a paletted PNG
Posted by mirek on Sun, 29 Oct 2006 23:34:44 GMT
View Forum Message <> Reply to Message

lindquist wrote on Sun, 29 October 2006 18:10I'm having some more problems.
I have this test code:

```
void WritePNG(const String& filename)
{
 ImageBuffer ib(256, 256);
 ib.SetKind(IMAGE_OPAQUE);

 RGBA pal[256];
 for (int i=0; i<256; ++i)
 {
  pal[i] = Color(i,i,i);
 }

 int idx = 0;
 for (int y=0; y<256; ++y)
 {
  RGBA* line = ib[y];
  for (int x=0; x<256; ++x)
  {
   *line = pal[idx%256];
   ++line;
   ++idx;
  }
 }

 PNGEncoder png(8);
 png.SetPalette(pal);
 png.SaveFile(filename, ib);
}
```

I would assume this code would generate a nice horizontal gradient, but it doesn't. Instead I get something that seems a bit like it's converted to 5bit colors. (every 8 pixels it jumps 16 indexes in

the palette)

That is unfortunately the limitation of palette conversion code - but in fact, we are not alone there... (almost any palette code I have studied before implementing U++ version did the same).

The problem is - you have 256 colors in palette and you need to find nearest color for 256*256*256 colors for tens of thousands pixels. If you would perform search for each pixel, you would be exporting each Image for hours.

Therefore you need something faster albeit less accurate. Usually, what you do is to cut some bits out and lookup index in map (you have to cut those bits because for full RGB you would need 16MB map). Works well except for single color gradients which is exactly the case you are trying....

Mirek

Subject: Re: saving a paletted PNG
Posted by lindquist on Mon, 30 Oct 2006 00:04:10 GMT
View Forum Message <> Reply to Message

That is not good news. What I really need is a true indexed ImageBuffer, not a indexed approximation of a RGBA ImageBuffer. Will I need to roll my own or is there a hack I could employ? I've been looking at the code but it seems the new stuff is very much RGBA.

Subject: Re: saving a paletted PNG
Posted by mirek on Mon, 30 Oct 2006 00:11:34 GMT
View Forum Message <> Reply to Message

Well, maybe the more detailed description of the problem could help me help you..

Mirek

Subject: Re: saving a paletted PNG
Posted by lindquist on Mon, 30 Oct 2006 00:18:46 GMT
View Forum Message <> Reply to Message

I have a terrain editor, which uses alpha layers for texturing. Each alpha layer has one byte per vertex: 0-transparent ; 255-opaque.
I'm trying to implement a export method that collapses the layers into a single indexed png with the pixel index being the material index. a material map.

vertices that have semi-transparent layers pick the least transparent one.

I would personally dump the bytes in a binary blob, but I have request for the indexed png :/

Subject: Re: saving a paletted PNG
Posted by mirek on Mon, 30 Oct 2006 06:33:23 GMT
View Forum Message <> Reply to Message

OK, I see 3 possible solution:

* Quick fix, without altering U++, little bit ugly

See PalleteCv class, Get. As you are able to set any palette to RasterEncoder, it is possible, based on PalleteCv conversion, to build special palette and

RGBA PaletteIndex(int index);

that will force those shifted values to produce specific index. Later I would add this to U++. Would be a bit slow, but not extremely slow.

* Indexed mode in PaletteCv and/or RasterFormat

Alter conversion routines to use some channel (most likely R as direct palette index. E.g. by using (PaletteCv *)1 in Write

* Perfect but complicated solution

Well, this one already happened in Raster - add support for Raw modes in RasterEncoder (RasterEncoder::WriteLine(byte *data)). This is far the most work and paradigm shift too.

Right now I would vote for the first option, in the same time I am afraid that last option would be nice to have too. But these are orthogonal.

Mirek

Subject: Re: saving a paletted PNG
Posted by mirek on Tue, 31 Oct 2006 10:48:16 GMT
View Forum Message <> Reply to Message

OK, the "perfect option" (posibility to send format specific (non-RGBA) data to .png) now in progress...

Mirek

Subject: Re: saving a paletted PNG
Posted by lindquist on Tue, 31 Oct 2006 14:31:23 GMT
View Forum Message <> Reply to Message

This is very good news I was studying on how to implement that, but if an official implementation

will arrive I can wait. I'd much rather focus on developing the application right now.

And so far Ultimate++ is doing a very good job at allowing me to do that. This is my first issue except for GLCtrl which I have customised a bit.

Subject: Re: saving a paletted PNG
Posted by mirek on Sun, 05 Nov 2006 19:27:46 GMT
View Forum Message <> Reply to Message

OK, RasterEncoder::WriteLineRaw is reality.

Let me know if it solves your problem

Mirek
