Subject: Drawing raw data to an Image / Draw object?
Posted by blueapples on Wed, 08 Oct 2008 05:42:12 GMT
View Forum Message <> Reply to Message

Hi, I am starting to use the Magick++ library (http://www.imagemagick.org/Magick%2B%2B/) with U++. So far, I have been able to link it in and actually make a few simple calls using the library. However what I cannot figure out how to do is draw the results of these calls using U++'s systems.

I see that DrawImage has several methods that allow drawing lines, rectangles, etc. What I can't find is a simple way to set individual pixels of an image. The Magick::Image object I am using to manipulate images exposes an array of structures that provide R,G,B, and A values. What I need is a way to "copy" these values to a U++ Image or Draw object in order to render the resulting image in the U++ application.

How would I do this?

Subject: Re: Drawing raw data to an Image / Draw object?
Posted by kodos on Wed, 08 Oct 2008 08:03:45 GMT
View Forum Message <> Reply to Message

Use ImageBuffer

http://www.ultimatepp.org/srcdoc$Draw$ImgTutorial$en-us.html Point #3
and
http://www.ultimatepp.org/src$Draw$Image$en-us.html

Subject: Re: Drawing raw data to an Image / Draw object?
Posted by blueapples on Wed, 08 Oct 2008 23:57:38 GMT
View Forum Message <> Reply to Message

Awesome! Thank you... I really need to read the whole manual I guess.

Subject: Re: Drawing raw data to an Image / Draw object?
Posted by blueapples on Thu, 09 Oct 2008 01:03:49 GMT
View Forum Message <> Reply to Message

This is what I've got so far. There's a pretty major problem though. The RGBA structure seems to only allow a very limited color depth, namely what can fit in 3 bytes. Magick++ (and really most image formats like JPG and PNG) allow many more colors than this makes possible. To get the conversion to work I have to use a rather crude instrument, quantumToByte(), which converts from Magick++'s 32 bit value to a byte. This makes high color images look just terrible... is there a way to use greater color depth with ImageBuffer?

```
byte quantumToByte(m::Quantum q)
{
 return q / 65535 * 255;
}


class Canvas : public Ctrl {
 Image drawImg;

public:
 typedef Canvas CLASSNAME;
 Canvas();
 void Paint(Draw& draw);
 void SetImage(m::Image& newimage, int x = 0, int y = 0, int zoom = 1);
};


/* Canvas::SetImage: Copies pixles for the current view to a new
 Upp::Image object to be rendered on the control. The zoom value
 is a magnification multiplier - 1 is for actual size pixels, 2 for
 double sized pixels, etc. */
void Canvas::SetImage(m::Image& image, int x, int y, int zoom)
{
 Rect rect = this->GetRect();
 int width = rect.Width() / zoom;
 int height = rect.Height() / zoom;
 int zx, zy;
 RGBA p;

 // Constrain the viewport to the max size of the actual image
 if(width > image.columns())
  width = image.columns();
 if(height > image.rows())
  height = image.rows();

 m::PixelPacket *pixel = image.getPixels(x, y, width, height);

 ImageBuffer ib(width * zoom, height * zoom);

 byte i = 0;
 float v;
 for(int y = 0; y < height; y++) {
  //RGBA *l = ib[y];
  i = 0;
  for(int x = 0; x < width; x++) {
```

```
   if(zoom == 1) {
    ib[y * zoom][x * zoom].a = 255;
    ib[y * zoom][x * zoom].r = quantumToByte(pixel->red);
    ib[y * zoom][x * zoom].g = quantumToByte(pixel->green);
    ib[y * zoom][x * zoom].b = quantumToByte(pixel->blue);
   } else {
    // This really should use some sort of ASM box routine
    for(int zx = x * zoom; zx < (x+1) * zoom; zx++) {
     for(int zy = y * zoom; zy < (y+1) * zoom; zy++) {
      //if(zy + y < height && zx + x < width) {
       //p = ib[zy + y][zx + x];
       ib[zy][zx].a = 255;
       ib[zy][zx].r = quantumToByte(pixel->red);
       ib[zy][zx].g = quantumToByte(pixel->green);
       ib[zy][zx].b = quantumToByte(pixel->blue);
      //}
     }
    }
   }

   // Go to the next pixel
   pixel++;
  }
 }

 Premultiply(ib);
 drawImg = ib;
}

void Canvas::Paint(Draw& draw)
{
 Rect rect = this->GetRect();
 draw.DrawRect(0, 0, rect.Width(), rect.Height(), Gray());
 draw.DrawImage(0, 0, drawImg);
}
```

Subject: Re: Drawing raw data to an Image / Draw object?
Posted by mrjt on Thu, 09 Oct 2008 12:35:12 GMT

View Forum Message <> Reply to Message

As far as I know there is no Image support for greater the 32-bit depths, since that is the effective limit of consumer display equipment.

But I believe your problem is that the conversion is wrong. I think it should be:
byte quantumToByte(m::Quantum q)

```
{
    // Equivalent to:
    // return (q * 255) / 65535);
    // or
    // return iscale(q, 255, 65535)
    return (q  >> 8);
}
```

For correctness I believe you should replace the 8 with '(QuantumDepth - 8)', but that's really just a guess from the ducumentation.

Alternatively you can #define QuantumDepth to 8 yourself to tell the library to use 32-bit and avoid the conversion. If you're only planning on drawing to the screen this seems like a wise move to also reduce memory consumption.