Chapter 13

Images

An image is a two-dimensional reproduction. We perceive the reality of the three-dimensional world around us through a two-dimensional reproduction projected on our retinas, and share them through photography. We also create images that reproduce our perceptions or reflect our imaginations as images through drawing and painting.

The vertex pipeline studied in the previous chapters results in an image. It converts the vertices of a surface mesh from their positions in model coordinates into their (viewport) positions in an output image. These vertices can be used to represent the shape directly in a vector image, consisting of strokes and regions, or can be further processed into a raster image consisting of a rectilinear array of pixels.

13.1 Vector Images

A vector image represents shapes as outlines, sometimes delineating filled regions. Vector images are describes by a sequence of strokes, specified by the position at the beginning of the stroke and the position at the end of the stroke, indicating that a line should be drawn between the points. The line can often be further stylized by specifying its color, thickness or dashing. As when specifying lines to a graphics library, it is often efficient to specify a line strip as a sequence of vertices that connect neighboring pairs of a list of screen positions.

Similarly, a line loop (which adds a stroke connecting its last vertex to the first vertex) can create a simple closed polygon so long as the segments do not cross. Such line loops outline a region, and some display methods can allow these regions to be filled, with a solid color, a gradient that smoothly interpolates two or more colors, or even the colors and details of a second image.

Interactive graphics began with vector displays. The first display device was a cathode ray tube (CRT). The screen of a CRT consisted of a layer of phosphor that would glow when a beam of electrons was directed
at it. A pair of electromagnets were placed beside the beam, one horizontally and one vertically. Voltages sent to these electromagnets would deflect the electron beam so it could be directed to hit a desired position on the phosphor screen. Hence, to draw a line from \((x_0, y_0)\) to \((x_1, y_1)\), the horizontal electromagnet’s voltage was set to something proportional to \(x_0\) and the vertical electromagnet’s voltage proportional to \(y_0\), then the horizontal voltage and the vertical voltage were simultaneously changed to values proportional to \((x_1, y_1)\), causing the electron beam to continuously trace a line along the phosphor screen from \((x_0, y_0)\) to \((x_1, y_1)\).

The same mechanism is also be used to generate light shows by rotating mirrors to deflect a laser instead of charging electromagnets to deflect an electron beam. Plotters employed vector display by holding a pen on a
horizontal moving axis while using rollers to move the paper vertically. The Etch-a-Sketch toy is similarly a vector display, though requires some skill to turn the knobs to get a straight diagonal line.

Some vector displays can retain their information, such as a plotter or some long-persistence phosphors, such that a vector image need only be drawn once. More commonly, as with most vector displays and laser light shows, the vector image, once displayed, rapidly fades and needs to be refreshed. An aspect of the human visual system called persistence of vision allows an observer to see an image even if its elements are constantly redrawn after fading away. Our persistence of vision lasts about 62.5 ms such that we start to perceive an image as flashing when it is refreshed less than about 16 times per second.

Hence vector displays maintained a display list, a data structure held in memory indicating the path of the beam and when to turn it on and off. The amount of information that can appear on a refreshed vector display is thus limited by the length of the display list that can be processed in at least 62.5 ms. In some displays, the length of strokes can also effect how much information can be displayed in a refresh cycle.

While we rarely see vector displays in modern form, the vector image persists. Its display list becomes a vector image file format that serves as a small program that instructs a processor how a vector image is drawn. Vector images are resolution independent. They are used for font descriptions, such as TrueType fonts, and for page descriptions such as PostScript and PDF, such that the same file can work with a variety of resolutions for both display and print. They are used for line art, such as the clip-art illustrations for slide presentations, so they can be resized as needed.
13.2 Raster Images

Anyone who has watched television, seen a movie at a modern cinema, or used a cell phone is familiar with raster images. These are images stored as a 2-D array of colors. They are usually displayed on a screen of left-to-right rows of pixels running top-to-bottom (or bottom to top), streamed in this order from memory. The memory is called a frame buffer, which is a contiguous section of memory space devoted to holding the pixels currently displayed.

The conversion of a vector image into a raster image is called rasterization, and the conversion of a raster image into a vector image is called (image) vectorization. Rasterization of line art is the same process as the rasterization of shapes (e.g. triangle meshes) in rendering in computer graphics. Image vectorization can be more complicated, because often one has to infer the original shape from the rasterized result, and recent approaches rely on machine learning techniques.

The number of pixels in a raster image or display is called its (spatial) resolution. The resolution of a display indicates how much memory must be devoted to its frame buffer, and the resolution of an image indicates how much memory is needed for its (uncompressed) storage. For example, a screen resolution of $1920 \times 1080$ requires storage of $1920 \times 1080 = 2,073,600$ pixels. If a pixel requires 3 bytes of storage (8 bits each of red, green and blue channels), then the screen’s frame buffer would need $2,073,600 \times 3 = 6,220,800$ bytes (just under 6MB) of memory. The amount of memory needed per pixel is related to the display’s color resolution indicating how many colors can be simultaneously displayed. Each pixel of a 24 bpp (bits per pixel) image can be one of $2^{24} = 16,777,226 \approx 16M$ different colors, whereas an alternative 8 bpp representation packs 3 bits for red, 3 bits for green and 2 bits for blue into a single byte for a pixel that can represent only 256 different colors. (We perceive red and green wavelengths better than blue, which is why blue gets only two bits.)

The shape of a raster image is usually a rectangle. The aspect ratio of the image (or display) is the horizontal size divided by the vertical size. For example, the aspect ratio of HDTV is 16:9, which means the ratio of its horizontal size to its vertical size is $16/9 = 1.7$. The pixels themselves also share an aspect ratio. For example, a common resolution for HDTV is $1920 \times 1080$, such that $1920/1080 = 1.7$. In this case, the pixel aspect ratio is 1:1, so called square pixels. If $A$ is the image/display aspect ratio, and $H,V$ are the horizontal and vertical resolution of the image/display, then the pixel aspect ratio is $(H/V)(1/A)$.

Once the last pixel is streamed from the frame buffer to the display, the whole process starts over again with the first pixel, which refreshes the screen and enables motion. (Originally this refresh was needed to recharge decaying phosphors.) The human perceptual system needs an update rate of 10Hz to perceive motion, thought we notice flicker up to
about 30Hz. Older technologies managed a reasonable update rate, such as film (24Hz) or NTSC television (30Hz). Modern displays can update at 60Hz, 120Hz or even 240Hz for even smoother motion, though this can introduce other undesirable perceptual effects [?]. This temporal resolution of the display is called its refresh rate. Storing an uncompressed video would require memory proportional to the product of the video’s spatial, color and temporal resolutions.

13.3 Color Depth

The number of colors that can be displayed in a pixel is another resolution, more commonly referred to as the color depth. The number of bits per pixel indicates how much information must be stored when the image is saved in an uncompressed format. Modern displays use 24 bits (3 bytes) of information to represent the color of each pixel, organized into three RGB channels: 8 bits for the red channel, 8 bits for the green channel and 8 bits for the blue channel. Since the human eye can resolve only about 100 shades of any hue, this is a reasonable discretization. This allows 256 levels of each color channel, and the combination of these leads to $256^3 = 2^{24} = 16,777,216$ different displayable colors, which also exceeds the 10M colors the human visual system can perceive (Judd, Deane B.; Wyszecki, Gnter (1975). Color in Business, Science and Industry. Wiley Series in Pure and Applied Optics (3rd ed.). New York: Wiley-Interscience. p. 388. ISBN 0-471-45212-2. from wikipedia).

When space is limited, the color depth of an image can be reduced. Another popular standard is to store 8 bits per pixel. This discretization can be organized into 3 bits of red, 3 bits of green and 2 bits of blue, based on the human visual system’s reduced capacity for blue hues which appear dimmer and focus off the retina due to the eye’s lens chromatic aberration.

13.4 Temporal Resolution

Raster displays can trade temporal resolution for spatial resolution by using an interlaced scan. An ordinary progressive scan display would display every line of each image frame of an animated sequence, whereas an interlaced display would display every other line of a first frame image, and fills in the skipped lines with those from a second frame image. Hence an interlaced display can display frames at twice the rate, but at half the spatial resolution.

For example, HDTV has both progressive and interlaced formats. A 60Hz 1080p format would display a full 1080-line frame 60 times per second, whereas a 60Hz 1080i format would display a 540-line frame 120 times per
second, by displaying the odd lines of the first frame followed by the even lines of the second frame.

Converting between progressive-scan and interlaced formats is straightforward for still motion scenes but can be tricky for fast motion. Displaying a progressive-scan signal on an interlaced display can result in stuttered motion, whereas displaying an interlaced signal on a progressive display can yield a dual edge effect called ghosting. A common approach for deinterlacing an interlaced sequence for progressive-scan display is called “bob and weave.” When image motion is significant, the “bob” algorithm is employed, which fills in each missing scanline with a copy of a neighboring line, or an interpolation between multiple neighboring lines. When image motion is low, then the “weave” algorithm simply fills in the missing scanlines with the corresponding lines in the next field, or an interpolation of these lines from previous and subsequent fields.

13.5 Gamma Correction

13.6 Transparency

13.7 File Formats

When writing a graphics program that generates an image on the screen, often it is useful to record that image in a file. While a huge number of
different image file formats exist, a few have become ubiquitous. Here are a few handy file formats that most programs can read, simplified for quick implementation in your graphics programs. It is easy to output a specific version of a file format, but much more difficult to read (or write) image files in all the various versions supported by these formats. Please see the specifications and supporting documentation for the complete file format descriptions.

PPM

The NetPBM format includes a very simple text file format for RGB raster images that makes it very easy to output a picture. In its simplest form it is the magic characters "P3" followed by the spatial and color resolutions followed by a stream of data, all text separated by white space.

```c
printf("P3 %d %d 255 ",width,height);
for (j = 0; j < height; j++)
    for (i = 0; i < width; i++)
        printf("%d %d %d ",red[j][i],green[j][i],blue[j][i]);
```

BMP

The Microsoft BMP (BitMaP) format represents a raster image as a sequence of RGB byte triples. While BMP is full featured, we’ll just focus on the simplest components need to store an uncompressed RGB file. This simple BMP format consists of a 14-byte header followed by a DIB header. All integers are stored in little-endian order with the least-significant byte first.
fput16(FILE f, short int x) {
    fputc(f,x & 255); x >>= 8; fputc(f,x & 255);
}

#define THRICE(x) x x x
fput32(FILE f, long int x) {
    THRICE(fputc(f,x & 255); x >>= 8;) fputc(f,x & 255);
}

int width, height;
int padded_row = ((24*width-1)/32 + 1)*4;
fputc(f,'B'); fputc(f,'M');
fput32(f, 54 + padded_row*height);
fput32(f, 0); fput32(f, 54); fput32(f, 40);
fput32(f, width); fput32(f, -height);
fput16(f, 1); fput16(f, 24);
fput32(f, ((24*width-1)/32 + 1)*4*height);
fput32(f, 0); fput32(f, 0); fput32(f, 0); fput32(f, 0);

for (int j = 0; j < height; j++) {
    for (int i = 0; i < width; i++)
        fputc(red[j][i]); fputc(green[j][i]; fputc(blue[j][i]);
        for (int padding = 3*width; padding < padded_row; padding++)
            fputc(0);
}

Figure 13.5: Code fragment for writing a BMP file.