Scanning Information

Acquiring Images

You have a lot of flexibility in how you acquire images for your multimedia titles. Alternatives range from purchasing images from an image bank or professional photographer to capturing images yourself. Costs, image quality, and licensing/copyright issues can affect your final decision. The following methods identify four common alternatives:

Purchasing—buying digital images
Creating—creating electronic images using specialized software
Scanning—digitizing photographs or flat art using a color scanner Digitizing
Video—digitizing video frames using a camera and frame-grabber

Resolution

Several kinds of resolution can affect bitmap quality: screen resolution, image resolution, and pixel resolution. You should understand the differences between all three when working with bitmap images.

Screen resolution is the maximum image area of the computer screen, expressed in horizontal and vertical pixels, for a particular video mode. The standard video mode for the Multimedia PC is 640 pixels by 480 pixels. You'll have to consider screen resolution when establishing a target image size for a scanned photograph.

Image resolution is the size of the digitized image expressed in horizontal and vertical pixels. The image resolution can differ substantially from the screen resolution. For example, say you display a 320-by-240 pixel image on a 640-by-480 pixel display. In this case, the image size is one-half the screen resolution, so the digitized image only fills one-half of the screen. When the image size and screen resolution are identical, the image fills the screen. When the image size is larger than the screen resolution, the screen can display only a portion of the image—requiring the display software to support scrolling to see other portions of the image.

Pixel resolution can become a factor when you move images between different graphic display modes or computer hardware. Pixel resolution refers to the ratio of a pixel's width to its height (also known as the pixel's aspect ratio). This can cause unexpected distortions in an image that's transferred between machines with

different pixel resolution
s. For example, if you capture an image on a device that uses rectangular pixels with an aspect ratio of 1:2, and later display it on a device that uses square pixels with an aspect ratio of 1:1, the image will be distorted. Fortunately, pixel resolution inconsistencies don't occur frequently as most displays use square pixels with an aspect ratio
of 1:1. Also, most capture devices let you adjust the pixel aspect ratio for your system.

Getting Better Results When Scanning

Here are four ways to get the best results with OCR:

- Quality of the original text
- Type of scanner
- Speed (power) of the computer
- Quality of the software

Start with High-Quality Printed Material
The process of optical character recognition is much like a person reading. Like a person's eye, the scanner analyzes light reflected from a page to create a pictorial representation of what is black (the ink) and what is not black (the paper). This picture of the page is stored in the computer where the OCR software tries to chop the black parts
up into individual letters and then guess what each one is.

Generally anything difficult for a person to read will be impossible for an OCR system to convert. This includes such things as smudged text, small type, strange fonts, characters that are too close together, and typewritten characters created with an old ribbon or dirty keys.

The bottom line to OCR efficiency: always start with high-quality printed material.

Choose an Appropriate Scanner
Given good clean text, the next critical element in the system is the resolution of the scanner. Typical scanner resolutions range from 75 to 450 dots per inch (DPI). The higher the resolution of the scanner, the higher the likelihood of accurate recognition by the software. Although some companies claim to be able to accurately recognize text at
200 DPI, 300 DPI is probably the lowest practical resolution.

Another factor that influences the suitability of a scanner for OCR is its method of feeding. The two standard methods are **flatbed and roller-fed**.

Flatbed scanners are like photocopiers in that the artwork to be scanned is placed on a glass plate, covered by a lid, and then passed over by a light.

Roller-fed scanners are like a typical FAX machine, in that artwork is scanned after being fed in through rollers.

Flatbed scanners are well-suited for bound, oversized, or especially small pages. Roller-fed scanners are good for bulk scanning of material with a consistent format. A flatbed scanner with an optional document feeder offers the best of both worlds. Use a Fast Computer.

The power of the computer used in an OCR system doesn't effect the accuracy of conversion. Nevertheless, we recommend the computer have at least an 80386 processor running at 25 Mhz. Also, since most OCR programs require large amounts of memory, having 4 MB of RAM or more can make life a lot easier.

Select Fast, Accurate OCR Software

There are dozens of OCR programs on the market. Most of these are simply software programs that you load into a computer and run, though some high-end programs work with a board that plugs into the computer. These hardware and software combinations are usually faster, more accurate, and more expensive.

There are a lot of options available when buying OCR programs. Some of the more interesting ones can do the following things:

- Read both mono and proportional spaced fonts
- Read dot-matrix output
- Learn new fonts
- Mix text and graphics
- Define frames of text to read
- Retain character formatting attributes Retain columns (tables)
- Save in various word processing formats Incorporate spell checking into the conversion process

Be aware, however, that character recognition systems rarely achieve more than 95% accuracy on anything but the cleanest of text. Even at 99% accuracy, there could be 20 mistakes on a typical 2,000-character page. So, budget time for spell checking and proofreading.

There are almost infinite combinations of specific scanners, computers, and software packages. If you have printed pages numbering in the hundreds, OCR may work for you. If you have massive amounts of printed text, you should consider hiring a data entry service to have it re-keyed.

Capturing from Video

Along with a scanner, you may want to capture images with a video camera. The camera focuses on the image and then transfers that image to the PC through a special interface board that converts the analog video signal to a digital format. This digitizer simplifies the image by combining palette values and setting the resolution. It also converts the digitized image into a standard graphics format such as TGA or PICT, which can be easily converted by the MDK's Convert tool. This setup lets you grab images with the video camera that can be read and enhanced by software the same as a scanned image.
Many different types of cameras exist and they can deliver images in many different formats (such as direct video, composite video, and National Television Standards Committee (NTSC)). After you've captured the image, you still must convert and enhance the image for incorporation within your application. The following paragraphs summarize the steps involved in this process.

Note:
A video camera usually digitizes images faster than a scanner, but it doesn't necessarily provide the best quality for the money. So, unless you're willing to get the best equipment, you might be better off buying a scanner for your basic image capturing.

Set up Your Copy Stand and Lights Properly
A proper setup and lighting can cut hours of image processing time. Using a copy stand as it comes out of the box invites trouble because quite often the lights are set to light the center of the stand. Often, one strong light (say, 1000 watts about five feet from the picture), will give you the proper, even illumination levels. Sometimes it is quite effective to tack the image to the wall and shoot it using a tripod-mounted camera.

Connect the Camera to the PC
To digitize images, you must plug a digitizer board into your computer, install the digitizer software, and connect the camera to the PC through the board. You may also want an extra monitor for viewing the camera image and some test equipment to ensure the best quality. Make sure all video equipment used is properly calibrated for color accuracy and reproduction.

Digitizers only accept RGB input, so you'll need to ensure your VCR or camera can output RGB. If you are capturing an NTSC signal, you'll need to convert it to an RGB signal first by using a special NTSC decoder.

Use a video camera whenever possible as it provides a more stable and higher resolution signal than a VCR. The next best signal is usually a live feed from cable, broadcast TV, or videodisc. Use VCR images only as a last resort. They produce highly unstable signals and low resolution output (around 240 lines of resolution).

Capture Frames
Focus the camera on the object(s) whose image you want captured. Use the video capture hardware and software to grab the image and store it to disk. All frame grabbers tend to distort the image slightly due to motion. The frame grabbing software is designed to compensate for this to some degree, however, for best results use still or slow moving images. Avoid grabbing in pause as the image is the most unstable in pause mode.

If you are capturing an entire sequence of video frames for an animation sequence, you'll have to shrink the original frames to a size that can be played back at a particular rate by the hardware. The standard Multimedia PC can play back at approximately 10-15 frames per second, with a frame size of approximately 100 pixels by 150 pixels by 8 bits. Reducing the frame size is probably best done with the original videotape before digitizing since the entire sequence will then be a consistent size. Some video boards can shrink a full screen video source. Otherwise, you may need to hire technicians in a TV studio at a greater cost.

Correct the Aspect Ratio
Your targeted aspect ratio is the VGA monitor at 640:480 (or 4:3). The resolution of your camera or video image is bound to be different so you'll want to be sure the capture software you use can convert and correct the aspect ratio.

Enhance the Image
You'll need to crop the image to size and perform any necessary color, edge, or contrast enhancements. Edit the image to clean up any problems introduced when captured. Use the greatest color or monochrome depth possible. For instance, when touching up a 24-bit color image, use a software package capable of handling 24-bit colors. The Windows with Multimedia tools only handle 8-bit color manipulation.

Scale the Image
By scaling the image down to less than full screen, you won't notice the lower resolution of the original video image. If you capture multiple video frames for an animated sequence, you'll have to shrink the original frames to a size that can be played back at a particular rate by the hardware. Contact a television studio for details as this requires professional expertise.

Transfer Images to PC
If you capture digital images on a different computer than your multimedia application development system, you'll need to transfer your images to the development platform.

Collect and Register Resources

You can gather data resources from many sources. You can get images from clip art, scanned photographs, or original computer artwork. Audio can be created, copied, or taken from analog/digital recordings. And tremendous volumes of text exist in multiple formats. There's no shortage of available resources.

The problem is sorting out what you need from what you don't. Using a DBMS to register resources as they're collected can be extremely helpful.

You should design your database to ensure it contains the information about each resource you want to record. Details associated with each record may depend on the type of resource registered. For example, the database record for an audio resource may include whether it's been recorded at 22.05 kHz or at 44.1 kHz. The database record for an image resource might instead include the number of bits used for the image depth.
Another area helped by the use of a DBMS is tracking data resources built from parts of several different resources. This situation will not be uncommon. Your database design should let you register the new resource into the database. For example, say you register an image showing all U.S. presidents. You then use an image editing package to pull
out all presidents elected during the twentieth century. You now have a new image to enter in the database. You may want your database to identify the relationship between the original image and its derivative image.

Convert the Text

Once you have the text tagged, you then must convert it to your target delivery format (e.g., ASCII or RTF). The many different word processing file formats has generated many different conversion programs. For example, Word for Windows supports the conversion of many different word processing formats into RTF and ASCII. Many word processors also provide utilities to either convert text to RTF or they let you save text in ASCII format. (Understand, however, that you lose all formatting when you save to ASCII.)

All methods of text format conversion involve pattern recognition of some sort. The source file is scanned until a significant pattern is spotted, and then some appropriate action is performed—typically deleting the pattern or replacing all or some portion of it with another pattern. You should investigate the following types of conversion programs:

Specific purpose—these programs automatically convert from one specific format to another. An example would be a WordStar to Word conversion utility. To run one of these, you simply provide the name of the input and output files and let the software run. If you can find or write one of these programs that matches your needs, it will proba

bly be the fastest way to make your conversions.

General purpose—these programs use some sort of look-up table to make conversions. By editing the look-up table, you can control the pattern searched for and what its replacement will be. These programs can support extremely elaborate patterns, but are only as good as your ability to initially recognize and uniquely define those patterns.

Word processors with Search and Replace—a powerful word processor with a strong macro language can often be useful during some stages of text conversion. But it can also be slow and cumbersome if you are working with megabytes of text. Microsoft Word for Windows is an especially powerful example of this category.

Creating Bitmaps in a Program

Windows includes five functions that let you create a device-dependent GDI bitmap object in your program. The first is the CreateDIBitmap function. The others are:

```
hBitmap = CreateBitmap (cxWidth, cyHeight, nPlanes, nBitsPixel, lpBits) ;
hBitmap = CreateBitmapIndirect (&bitmap) ;
hBitmap = CreateCompatibleBitmap (hdc, cxWidth, cyHeight) ;
hBitmap = CreateDiscardableBitmap (hdc, cxWidth, cyHeight) ;
```

The CreateDiscardableBitmap function is rarely used and is not recommended. In all cases, the cxWidth and cyHeight parameters are the width and the height of the bitmap in pixels. In CreateBitmap, the nPlanes and nBitsPixel parameters are the number of color planes and the number of color bits per pixel in the bitmap. At least one of these parameters should be set to 1. If both parameters are 1, the function creates a monochrome bitmap. (I'll discuss how the color planes and color bits represent color shortly.)

In the CreateBitmap function, lpBits can be set to NULL if you are creating an uninitialized bitmap. The resultant bitmap contains random data. In the CreateCompatibleBitmap and CreateDiscardableBitmap functions, Windows uses the device context referenced by hdc to obtain the number of color planes and the number of color bits per pixel. The bitmap created by these functions is uninitialized.
CreateBitmapIndirect is similar to CreateBitmap except that it uses the bitmap structure of type BITMAP to define the bitmap. The following table shows the fields of this structure:

Field	Type	Description
bmType	int	Set to 0
bmWidth	int	Width of bitmap in pixels

bmHeight	int	Height of bitmap in scan lines
bmWidthBytes	int	Width of bitmap in bytes (must be even)
bmPlanes	BYTE	Number of color planes
bmBitsPixel	BYTE	Number of color bits per pixel
bmBits	void FAR *	Far pointer to array of bits

The bmWidthBytes field must be an even number—the lowest even number of bytes required to store one scan line. The array of the bits referenced by bmBits must be organized based on the bmWidthBytes field. If bm is a structure variable of type BITMAP, you can calculate the bmWidthBytes field by using the following statement:

```
bm.bmWidthBytes = (bm.bmWidth * bm.bmBitsPixel + 15) / 16 * 2 ;
```

If Windows cannot create the bitmap (generally because not enough memory is available), it will return a NULL. You should check the return values from the bitmap creation functions, particularly if you're creating large bitmaps.

The handle to the bitmap is not a handle to a global memory block, so don't try to use the GlobalLock function on it. The handle is instead a local handle to the GDI module's data segment. This handle references a small local memory block in GDI that contains a second handle to a global memory block containing the information in the BITMAP structure and the actual bits.

Once you create a bitmap, you cannot change the size, the number of color planes, or the number of color bits per pixel. You would have to create a new bitmap and transfer the bits from the original bitmap to this new bitmap. If you have a handle to a bitmap, you can get the size and color organization using:

```
GetObject (hBitmap, sizeof (BITMAP), (LPSTR) &bitmap) ;
```

This copies the information about the bitmap into a structure (called bitmap here) of type BITMAP. This function doesn't fill in the bmBits field. To get access to the actual bits of the bitmap, you must call:

```
GetBitmapBits (hBitmap, dwCount, lpBits) ;
```

This copies dwCount bits into a character array referenced by the far pointer lpBits. To ensure that all the bits of the bitmap are copied into this array, you can calculate the dwCount parameter based on the fields of the bitmap structure:

```
dwCount = (DWORD) bitmap.bmWidthBytes * bitmap.bmHeight *
                  bitmap.bmPlanes ;
```

You can also direct Windows to copy a character array containing the bitmap bits back into an existing bitmap using the function:

SetBitmapBits (hBitmap, dwCount, lpBits) ;

Because bitmaps are GDI objects, you should delete any bitmap you create:

DeleteObject (hBitmap) ;

End.