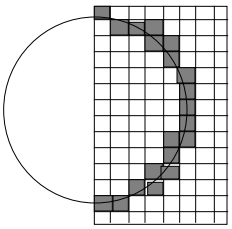
A R C H 3 7 0

CompuFacts #4

Revised: 5.January.1997

Computer Graphics
A Tale of Two Technologies
Raster vs. Vecctor
Consequences
Dr. Pixel & Mr. Hide
Summary



Objects? or Pixels?

Two Technologies

Two fundamentally different techniques exist for representing and manipulating graphics in computers. One way to understand them is to think about the two ways we might use a telephone system to transmit a graphic from one city to another.

One way would be to pick up the graphic in one had, and the phone in another and simply tell the person on the other end what the graphic looks like–*"In the middle of the page there is a 3" diameter red circle, divided in half horizontally, with a heavy green line. The lower half of the circle is filled in with vertical black dotted lines spaced 1/8" inch apart, etc"*

The other way would to connect both phones to fax machines and simply feed the page through the machine on your end. This approach uses a very simple, information-poor, highly mechanistic technique.

The first technique is the *object oriented* or *vector* approach. It depends on a possibly rich vocabulary of geometric shape names, modifiers such as color and line-type, coordinate or positioning information, and so on. It is capable of quite precise and detailed descriptions, though they may take some time to create and transmit, and some of the "personality" of a hand drawing may well be lost in the process.

The FAX machine, on the other hand, is a great example of *raster* technology. It scans across and down the page, dividing it into a grid of little squares. Each square is examined and identified as "black" or "white". A stream of information ("black", "black", "white", "black", etc.) is sent to the remote machine which colors in dots on a blank piece of paper so that it closely resembles the original. If the original contains irrelevant coffee-cup stains or finger-print smudges, they are treated as having the same validity as anything else on the page. No vocabulary, no precision, and only an approximation of the original, but often perfectly adequate.

Other names are sometimes used to refer to these two information technologies. The terms *draw, object-oriented graphic, vector graphic*, all tend to refer to the vocabulary-oriented description of the graphic, while the terms *raster, pixel-graphic, paint graphic*, and *bit-mapped graphics* all refer to the "sampled", or fax-like technique.

In vector/draw/object representations we store and manipulate a *description* of the graphic. Such a description would use concepts like "line" and "circle" to convey information about the geometrical elements (or primitives) of which the larger graphic is composed. A raster/paint/pixel representation, on the other hand, is more like a fax transmission of the graphic. It "colors in a grid" in *approximate* representations of the primitive shapes. As you might expect, describing some graphics is much harder than just faxing them, even though the fax is jagged around the edges.

While it may seem like they are largely comparable, there are actually a number of differences, because we aren't just transmitting a graphic from one city to another, we're converting a graphic from an idea in our heads (or a paper graphic) into data in a computer, and then printing it out again as a graphic on paper. Two data-conversions are therefore possible.

Vector vs. Raster

The most fundamental distinction that exists in computer generated graphics is that between *vector* and *raster* graphics. This distinction, which was outlined above, derives from the two different ways (or "imaging technologies") which exist for forming pictures on output devices, whether computer screens or pieces of paper, but its effects are so fundamental that they are seen in the ways in which *programs* operate on graphics, and therefore in the kind of *data* which is stored and manipulated by the programs.

The visible distinction between vector graphics and raster graphics has become somewhat blurred in recent years, especially with the dominance of laser and ink-jet printer technology and windows operating systems (such as the Mac and Windows OSs). All of these are rooted in raster technology. At this time most printers and plotters and almost all CRT screens are raster displays. Thus, the visible artifacts of computer graphics are almost invariably raster. None-the-less, object-oriented *data* and *editing* operations are a critical part of architectural computer use. It is important to "see behind" the raster output devices and see how the graphic is created and maintained by the computer program.

Since the distinction derives from different display technologies, lets look at these two categories of devices, keeping in mind that the difference between the two types of device have created differences in data and program as well.

Vector technology

Vector technology is used for CAD and drawing programs. Vector *devices* are those which are able to draw straight lines between any two points. People are vector output devices. We draw, or stroke, each line of the image, in no particular order, building up the image over time. Vector representations

involve information about lines–endpoint coordinates, colors, linestyles, and so on.

The simplest vector representations involve only lines but more complicated ones might include circles, text, etc. These elemental drawing components are called *primitives.* Pen plotters are also vector devices. Neon-sign artistry is vector graphic artistry.

primitives

Programs that draw using vector devices describe each line in terms of the coordinates of its end-points, circles in terms of the center and radius, etc. For this reason vector graphics are sometimes said to be described by equations. This is true in the sense that the drawing program maintains a database describing the image in terms of shapes with mathematical descriptions. For example, the arc on the left side of the image at the top of the first page might be described as "Arc,0,0,1, 0,180" meaning "arc centered at 0,0, radius of 1, spanning from 0 degrees to 180 degrees". This does not mean that the user enters an equation, just that the graphic produced by the description adhears to an equation.

draw programs

Programs that manipulate vector data, (often called *draw* programs or *object-graphics* programs) store a description of the graphic as a list (sometimes called a display list) of various "things to be drawn" (hence the name "object-graphics:" the "things" are called objects–it sounds better). Editing the graphic is done somewhat indirectly by editing the database that describes it. The program then redraws the graphic.

Examples of programs in this category include AutoCAD, Freehand, Illustrator, MacDraw, and ClarisCAD.

Raster technology

pixels

Raster technology began with satellite images, but is most visible today as the technology behind "paint" programs. Unlike vector displays, raster devices "draw" lines by "turning on" or "coloring in" dots (called *picture elements*, or *pixels*) in a grid. This produces an *approximation* of the line which we see on the screen. Most Time/Temperature signs are raster devices, as are FAX machines, and all the displays that you have seen in this class as well as most of the output, including laser printers.

rasterization

The concept of a "line" is not central to raster data, though it's fairly easy to "overlay" an object representation on a set of pixels as a preliminary step to converting the object graphic into a raster graphic. This is illustrated on the right side of the circle graphic on page 1. The right hand arc is still visible on top of the grid and it's approximate representation of the arc. The process of converting data from an object description into a raster one is called "rasterization" and explains how we can have displays of "object" data on raster CRT screens or laser printers.

It's important to understand that in a paint program the object data is not stored as part of the finished raster image. Once the "circle" has been rasterized, the program discards the information about it's center and radius, or even that there is a circle! It is just one way to turn ON or OFF a bunch of pixels.

Raster data in a black and white system consists of a series of 0's and 1's, where the 1's represent the black pixels and the 0's the white ones. They implicitly describe the colors of pixels in a rectangular grid, filling in from left to right and top to bottom.

The rasterized arc on the right side of our key image would be stored as 10000000011100000001100000001000000001000-0000010000000100000001000000011000000100000011000-0011000011000000000000000. That's quite a bit different from the object-oriented description: "Arc,0,0,1, 0,180"

Consequences

Complexity and time

more complexity = more time

One very important characteristic of **object graphics** can now be understood. That is, more lines in the drawing mean more data to process, so very complex drawings may take a long time to display and edit.

more complexity = same time

In contrast, the visual complexity of a **raster image** doesn't change the amount of information needed to display it. A blank image (all white or all black) contains just as much data as a complex one.

The Jaggies

"The jaggies" and resolution

There is generally a difference in visual quality between true vector and raster output: think about the difference between a neon sign (a vector display) and a Metro bus' destination sign or a "time and temperature" sign (both raster displays). The bus sign's characters are blocky and rough around the edges, a raggedness called *the jaggies.* One "cure" for the jaggies is to increase the number of *pixels* or dots used to represent the character. The number of pixels (per inch if you are discussing paper, or across the whole screen for CRTs) is called the *resolution* of the device. Thus, ***sufficiently high resolution raster devices can give the visual appearance of a vector device***. The problem is that such resolutions require very large amounts of memory to store and process the graphics, and that memory is expensive. A laser printer is a raster device, just like the bus' destination sign, but it is able to "display" the image at a much higher resolution. In fact, laser printers often have more memory and faster processors than the computers to which they are attached!

Editing

In an object oriented program it is possible to select a single object from the data base by comparing the users's input coordinates (from the mouse) with the coordinates of the various primitives. With a little math and an obvious test ("pick the object closest to the input coordinates") the user can

then "point at things" on the screen and the program can tell what they are. Further, in the case of our "arc", once the object has been selected, it would be possible to change any of the critical values which describe it–the center, the radius, the starting or ending angle.

paint programs bitmap

Raster programs, in contrast, don't know about the picture as a collection of lines, they only know about the dots or *pixels* on the screen, which are stored in the program as a *bit map* or *pixmap* (short for pixel map). In this sense, they store the image itself, rather than a description of the image. There are advantages to each, depending on what kind of a graphic result you are trying to produce.

MacPaint, PC Paintbrush and PhotoShop are pixel-oriented programs. AutoCAD, Microstation and Form•Z are object-oriented programs. Some programs (SuperPaint and Canvas for example) offer features of each. While programs may offer features of each, the operations (drawing tools, editing tools, etc.) available will be segregated. That is, you cannot do vector things to raster data, or vice versa. (of course, there are the occasional exceptions to the rule....but none you are very likely to see.)

images

Graphic images from pixel-oriented programs we will call "images" or "pixmaps" (pixel maps), while we call graphic images from object-oriented programs "drawings". In vector representations, we use concepts like "line" and "circle" (in 3D object-oriented programs these become "plane" and "sphere") to declare or store information about the model. A raster representation, on the other hand, "colors in the grid" for approximate representations of the primitive shapes (which the raster system does not, and cannot store). We can move, to a limited extent, back and forth between the two worlds, but only with sometimes irritating compromises. You will need to distinguish, in your future computing activities, between programs using object-oriented (line drawing) data, and those using pixel data. You will need to know what you can do with a file of pixel data, as opposed to the "same" graphic in object representation.

Hybrid programs

mixed mode paint/draw programs

While paint and draw data are quite different from one another, it won't always be a simple distinction to make. Most painting programs, for instance, present you with a variety of object-like "pseudo primitives" in addition to the pixel tools like the *spray can* and the *paint can*. The apparent line drawing tools are actually part of the program's "user interface"–they don't actually create line-drawing primitives in a drawing database. We might say that the program is performing a "sleight of hand"–presenting an *object behavior* as a transitory mechanism for acquiring *raster data*. It retains the line as object data just long enough to convert it to it's raster form (a process called rasterization) and then it discards that information. The rasterization process converts the line to pixels by "turning on" or "coloring in" the individual dots needed to represent the line. Once complete, it never remembers how any given dot got to be the color it is.

pixmap primitive

In a similar way, fundamentally object-oriented programs do sometimes offer a "pixel image" or "PIXMAP" primitive. Since a pixel image is simply dots in a rectangular grid, we can turn the image as a whole into an object by placing the (pixel) image onto an (object) rectangle, creating a small "rubber sheet" bitmap on the rectangle. We can then move the rectangle around, resize it, place it in front of or behind other objects, etc. We *cannot* edit the image on the rectangle with drawing tools. It's like a piece of wall-paper.

As we have already discovered, there are two main branches in computer graphics, raster and vector, sometimes called by the somewhat more general terms, "pixel" and "object". Understanding the distinction between the two is fundamental to understanding most graphics programs and "graphic events" like erasing or rendering or printing. Graphics data and software is divided between "painting" (pixel) and "drawing" (object) types. Graphic images from pixel-oriented programs we will call "images" or "pixmaps" (pixel maps), where we called graphic images from object-oriented programs "drawings". In vector representations, we use concepts like "line" and "circle" (in 3D, "plane" and "sphere") to declare or store information about the model. A raster representation, on the other hand, "colors in the grid" for approximate representations of the primitive shapes (which the raster system does not, and cannot store). We can move, to a limited extent, back and forth between the two worlds, but only with sometimes irritating compromises. You will need to distinguish, in your future computing activities, between programs using object-oriented (line drawing) data, and those using pixel data. You will need to know what you can do with a file of pixel data, as opposed to the same graphic in object representation.

Note! it is perfectly possible for a vector drawing *program* to display a picture on a raster *device*. Imagine placing a piece of grid paper over a line drawing and filling in each square of the grid paper behind which a line of the drawing is visible, as illustrated in the figure at the top of the first page. This process of converting from vector data to raster data is called *rasterization* . It is very easy to do, especially for a computer. In fact, it is a key element in almost all modern desktop or workstation computer systems.

Note also that in contrast to rasterization, converting from raster to vector data (which is called *vectorization*) is quite difficult. In a simple sense, this is because each pixel covers a certain amount of area and it is therefore very difficult to deduce through what part of the pixel the line (or is that a circle, or perhaps part of a letter?) passes.

Paint program vs. Drawing program: different personalities

Primitives are kind of like atoms. Since a primitive in a 2D drawing program is the fundamental, smallest thing of which a drawing is made, and a line is a primitive, you cannot really erase just part of a line. It's almost like the drawing was made by stretching rubber-bands (lines) between pins (end-points) stuck in the drawing. It's easy to move one of the end points, but to make a hole in the middle we need to actually insert a new pin/rubber-band/pin into the drawing, and shorten up the

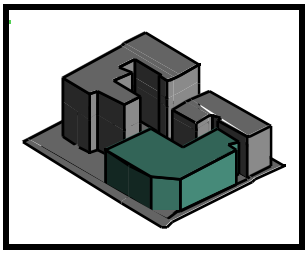
original one. In the CAD system you have to break the line into pieces (which inserts new 'rubber bands' into the drawing) and then delete the piece in the middle.

By contrast, in a paint program you would do this by simply swiping the eraser across the line as you would in a pencil-and-paper drawing. That's because the individual pixels are a lot like a pencil's individual instantaneous 'marks' on the page. In a raster system, erasing is simply the process of changing a pixel's color back to that of the background!

You may be thinking that a pixel-oriented system sounds like the smart way to do things, but consider how hard it is to erase one of two lines which are drawn quite close together. Wouldn't it be easier to simply "pull out the pins" for this rubber-band line, making it totally disappear without any danger of damaging the adjacent line? And what about moving just one end of a line? That's easy in a drawing program, but impossible in a paint system (you can't move one "end" of something that doesn't exist as discrete data!).

Again in a drawing program, unlike painting programs, if you have two overlapping objects and you move one of them away, the other will still be intact! It's like having a stack of paper drawings and sliding them around on the drawing board. That's something paint programs can't do, whereas drawing programs can, because the drawing program uses object-oriented data.

Including Images in Text Documents: Image Transfers, Manipulation & Printing



Quite often we will want to include images as part of reports, resumes, flyers, posters, etc. When laying out such a document, there are several issues that we will need to consider.

Let's say we have created a 3D model of a proposed project. From this model we are able to project 2D images, and we would like to use several of these images to illustrate a report. Since the 3D drawing, like a spreadsheet display, is only one representation of the model (like a photograph is one of many possible photographs of the same reality) the first thing to realize is that what we want to transfer is the 2D IMAGE, not the 3D DATA from the modeling program.

Most programs have an "**EXPORT**" or "**SAVE AS**" command (in the **File** menu). This will save the 2D screen images to PICT files (the most common Mac graphics-file format). While this is good, we will learn that the rendering algorithm chosen determines what *kind* of PICT file we will generate, raster or vector.

☛ It is unfortunate that the SAVE-AS "FORMAT" pop-up-menu, which offers several choices in addition to PICT, makes no distinction between which of those options pertain to saving a 2D *image* and which are saving 3D *data*, which can be quite confusing! DO remember to save your data as normal form•Z data too!

PICT[1]s: Raster or Vector format?

☛ PICT stands for "picture". The PICT file format, which is defined by Apple Computer, is the format used to store graphics on the Macintosh clipboard. This is why it is so commonly accepted by Mac programs.

PICT files

Macintosh PICT files can contain either purely raster or object data, or a mixture of the two. The contents are determined by the program which generates the PICT.

The modeling program may have several mechanisms (or *rendering algorithms*) which can be used to generate the image, some using raster logic, and the others using object or vector logic. The **DISPLAY** options for renderer select among the choices, but activating shadows (for instance) might cause a switch from an object to a raster rendering algorithm. Based on what kind of data the rendering algorithm generates, the **SAVE AS** formatting option PICT will create **DRAW** or **PAINT** data. The only way you can know for sure what you will get is to experiment a bit.

Picking your PICT type

The type of PICT image that you will need for a particular task will depend on that task as well as the complexity of the 3D model, the kind of final output or display and so on. Here are some of the issues.

Reasons to pick DRAW:

DRAW

- **Editability**
  The resultant PICT image will be composed of objects like lines, polygons, etc. You will be able to open that image in a drawing program (SuperPaint, MacDraw, Canvas, etc.) and modify it. You might remove some polygons to make a "cutaway" drawing. You might change the color or pattern or line-weight attributes of certain polygons.
- **Scalability**
  Because the image is described in terms of objects with coordinates, the entire image can be scaled up or down nicely, without nasty raster effects (like jaggies, lost data etc.)
- **Print Quality**
  Again, because of its object-based description, when the image is printed to a laser printer (or other higher resolution device, such as a slide camera, etc.), you will get the maximum benefits from that higher resolution.

Reasons to pick PAINT:

PAINT

- **Editing options**
  The resultant PICT image will be composed of a single

[1]Notice that PICT is four characters long and recall that there are two four-character tags on all Mac files, one of which indicates the file's creator, and one of which indicates the file's type. PICT is actually a Mac file type.

"object" called a PIXMAP; i.e., a pixel image. This file could be edited in ways the object-based image might not. Soft textures and shadows could be applied via tools like the paint-brush and the spray-can. Portions of polygons can be easily erased or re-colored.

- **Editing speed**
  Selecting objects in a large object-based image can take painfully long (see below), whereas complex pixel based images take no longer to edit than simple ones, so pixel data might simply be the only viable choice.
- **File Size and Complexity**
  One of the biggest reasons to pick PAINT is its lower image complexity and file size. An object-based drawing consisting of many polygons may be awkward to manipulate and edit if the machine you are using isn't fast enough, and 3D data often consists of lots and lots of polygons (that is, *no* machine may be fast enough). A pixel-based drawing, on the other hand, consists of simply the final pixel colors for each pixel, so model complexity has no effect on the image file size or complexity.

Converting Color Images into black and white data.

After generating an attractive color image, it can be very disappointing to see what happens to it when it is printed on a black & white printer. There may be cheap color printing available sometime, but it probably won't be in the very near future. Up until very recently it required quite a bit of skill and expertise to convert color images into black and white (called "**half-toning**"). While it is now easier, the results are not always satisfying. Word will happily import and display color PICT images on the screen (if you have a color monitor), and newer laser printers do a better job of half-toning, but you will not always have these available.

The free Mac image-editing program **IMAGE** (it should be in the Applications folder on Hard Disk) can be used to edit raster or object PICT files (the results are always a raster file). While a great many edit and transformation features are available (some of which do awful things to images!), there are three that you might be interested in:

Scale and Rotate...
in the **Edit** menu, this command might be used to pre-scale the image, before other transformations on it. This might yield a better-looking image when viewed or printed.

Convert to Gray scale
located in the **Enhance** menu, this option converts a multi-color image into a gray scale image, but still with multiple shades of gray.

Dither
also in the **Enhance** menu, this command converts an image into simple black and white pixels (i.e., it half-tones the image).

Be careful of the order in which you apply these commands, as it makes a very big difference in the appearance of the final image (recommended use is in the order listed above). Also, you might want to use the "paint bucket" tool to pour "white" onto the background color in order to get rid of it. You might also explore the "Enhance Contrast" command, and others. Other editing can also be accomplished here (such as sketching in people, or adding notations with arrows and circles, etc.)

Printing Color Images.

It's not uncommon to see students print their first color images and receive solid black and white pages as output. The newer Apple printer drivers will do half-toning *if you ask them to*. This is done in the "Print dialog box", which allows you to choose between "Black & White" and "Color/Gray scale" printing. The images which follow illustrate the difference. (Note: the graphics on this page have been edited to illustrate what *would* happen.)

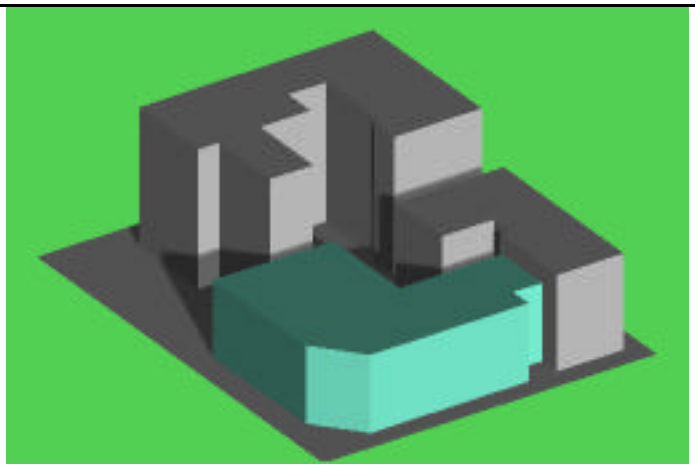
1

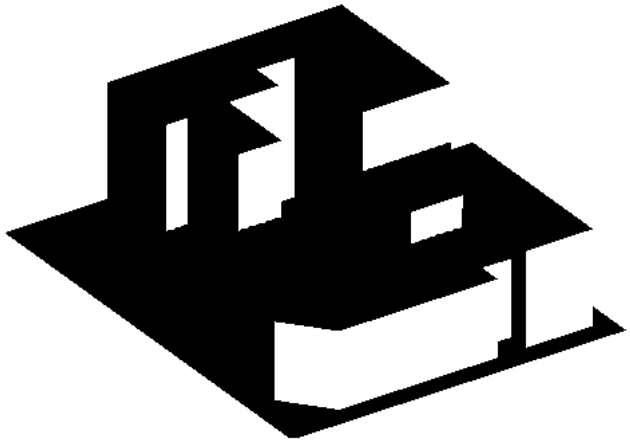
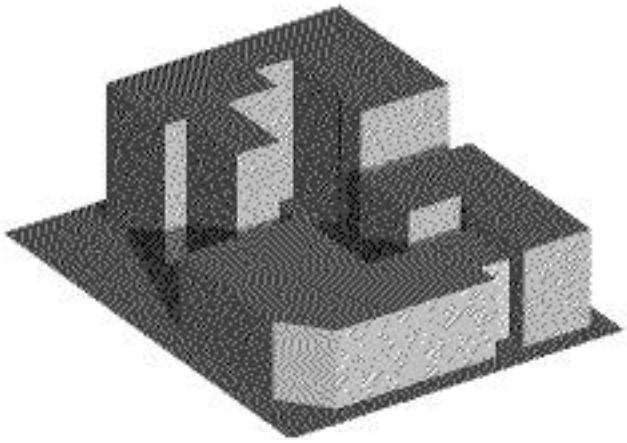
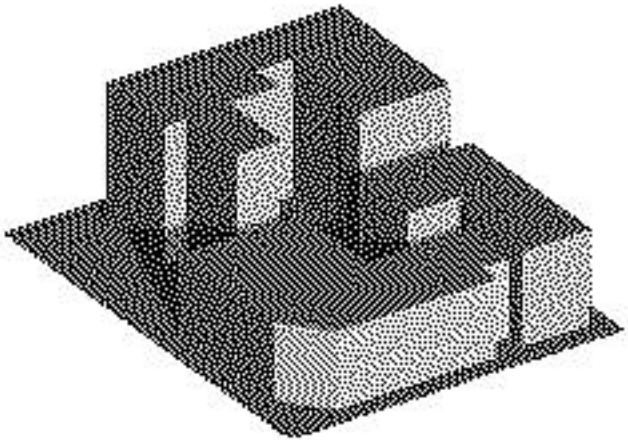
This is a COLOR raster PICT from Modelshop. It was inserted directly into Word, scaled to 50% and printed on a LaserWriter printer.

It is displayed in color on the Mac screen.

When "**Color/Grayscale**" is selected in the LaserWriter print dialog, you get a nice grayscale image.

On a color printer it would, naturally, print in color.



2 This is what image #1 would look like if printed with "**Black & White**" selected in the LaserWriter print dialog box. Of course, it would look the same on the screen *[This image has been edited, so it looks like the printed image would, not the screen image]*	
3 This is the same image after using the **Image** program to convert it into a Black & White PICT. It was then Inserted into Word and scaled to 50%. It looks the same whether printed to the LaserWriter in *Color/grayscale* or *Black & White*. It is almost solid black on the Mac screen since every screen pixel represents four image pixels (due to the 50% scaling)	
4 This is the same as #3, but the scaling to 50% was done in **Image** before it was inserted into Word. Fairly legible on the Mac screen, and prints the same using either *Color/grayscale* or *Black & White* option.	

Questions

Are humans vector devices or raster?

Is a FAX machine a vector device or a raster device?

Are laser printers vector or raster?

Vocabulary

pixmap

pixel

raster

vector

object

bitmap

paint

draw
primitive
Postscript®
pseudo-primitive