1. Application of Computer Graphics

Computer-Aided Design for engineering and architectural systems etc.

Objects maybe displayed in a wireframe outline form. Multi-window environment is also favored for producing various zooming scales and views.

Animations are useful for testing performance.

Presentation Graphics

To produce illustrations which summarize various kinds of data. Except 2D, 3D graphics are good tools for reporting more complex data.

Computer Art

Painting packages are available. With cordless, pressure-sensitive stylus, artists can produce electronic paintings which simulate different brush strokes, brush widths, and colors.

Photorealistic techniques, morphing and animations are very useful in commercial art. For films, 24 frames per second are required. For video monitor, 30 frames per second are required.

Entertainment

Motion pictures, Music videos, and TV shows, Computer games

Education and Training

Training with computer-generated models of specialized systems such as the training of ship captains and aircraft pilots.

Visualization

For analyzing scientific, engineering, medical and business data or behavior. Converting data to visual form can help to understand mass volume of data very efficiently.

Image Processing

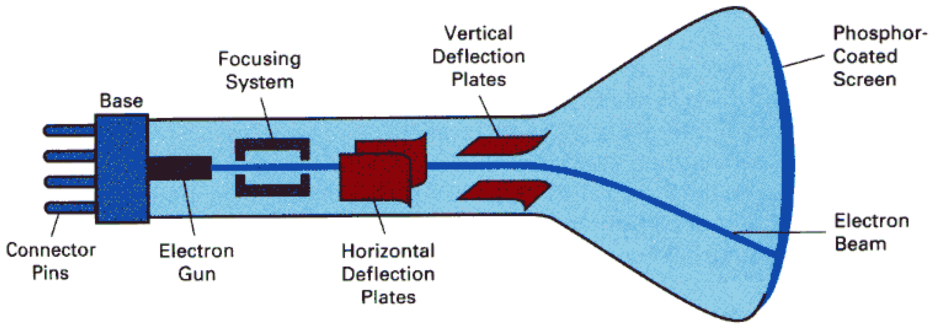
Image processing is to apply techniques to modify or interpret existing pictures. It is widely used in medical applications.

Graphical User Interface

Multiple window, icons, menus allow a computer setup to be utilized more efficiently.

2. Overview of Graphics Systems

2.1 Cathode-Ray Tubes (CRT) - still the most common video display device presently



Electrostatic deflection of the electron beam in a CRT

An electron gun emits a beam of electrons, which passes through focusing and deflection systems and hits on the phosphor-coated screen. The number of points displayed on a CRT is referred to as the

resolution (eg. 1024x768). Different phosphors emit small light spots of different colors, which can combine to form a range of colors. A common methodology for color CRT display is the **Shadow-mask** method.

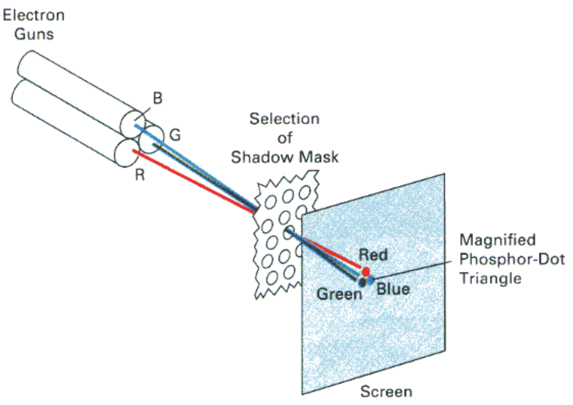
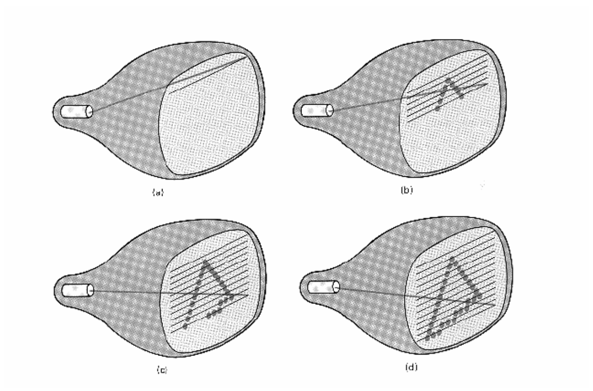


Illustration of a shadow-mask CRT

The light emitted by phosphor fades very rapidly, so it needs to redraw the picture repeatedly. There are 2 kinds of redrawing mechanisms: Raster-Scan and Random-Scan

Raster-Scan



The electron beam is swept across the screen one row at a time from top to bottom. As it moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. This scanning process is called refreshing. Each complete scanning of a screen is normally called a **frame**.

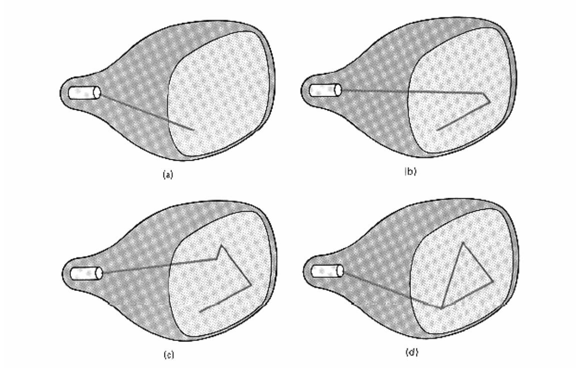
The refreshing rate, called the **frame rate**, is normally 60 to 80 frames per second, or described as 60 Hz to 80 Hz.

Picture definition is stored in a memory area called the **frame buffer**. This frame buffer stores the intensity values for all the screen points. Each screen point is called a **pixel** (picture element).

On black and white systems, the frame buffer storing the values of the pixels is called a **bitmap**. Each entry in the bitmap is a 1-bit data which determine the on (1) and off (0) of the intensity of the pixel.

On color systems, the frame buffer storing the values of the pixels is called a **pixmap** (Though nowadays many graphics libraries name it as bitmap too). Each entry in the pixmap occupies a number of bits to represent the color of the pixel. For a true color display, the number of bits for each entry is 24 (8 bits per red/green/blue channel, each channel 2^8=256 levels of intensity value, ie. 256 voltage settings for each of the red/green/blue electron guns).

<u>Random-Scan (Vector Display)</u>



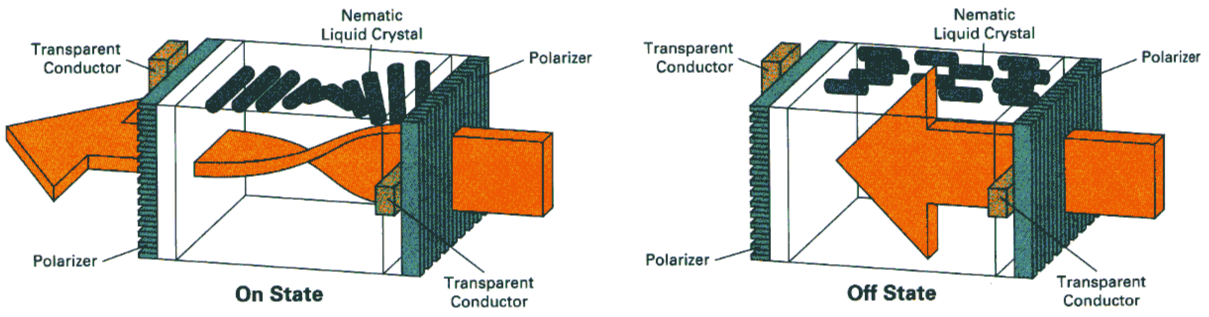
The CRT's electron beam is directed only to the parts of the screen where a picture is to be drawn. The picture definition is stored as a set of line-drawing commands in a refresh display file or a refresh buffer in memory.

Random-scan generally have higher resolution than raster systems and can produce smooth line drawings, however it cannot display realistic shaded scenes.

2.2 Flat-Panel Displays - will be the most common video display device very soon.

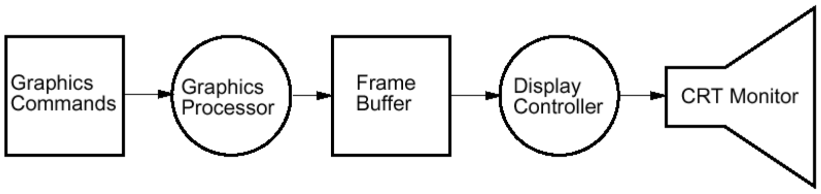
Reduced thickness, volumn, weight, and power requirements compared to CRT.

<u>Liquid-Crystal Displays (LCDs)</u>



- Liquid crystal refers to compounds which are in crystalline arrangement, but can flow like liquid.
- The light source passes through a liquid-crystal material that can be aligned to either block or transmit the light.
- 2 glass plates, each containing a light polarizer at right angles to the other, sandwich a liquid crystal material.
- Rows of horizontal transparent conductors are built into one glass page. Columns of vertical conductors are put into the other plate. The intersection of 2 conductors defines a pixel position. -- Passive-matrix LCD
- In the "on" state, polarized light passing through the material is twisted so that it will pass through the opposite polarizer.
- Different materials can display different colors.
- By placing thin-film transistors at pixel locations, voltage at each pixel can be controlled. -- Active-matrix LCD.

2.3. Graphics Systems



Block diagram of a CRT graphics system

In this context we discuss the graphics systems of raster-scan devices. A graphics processor accepts graphics commands from the CPU and executes the graphics commands which may involve drawing into the frame buffer. The frame buffer acts as a temporary store of the image and also as a decoupler to allow the graphics processor and the display controller to operate at different speeds. The display controller reads the frame buffer line by line and generates the control signals for the screen.

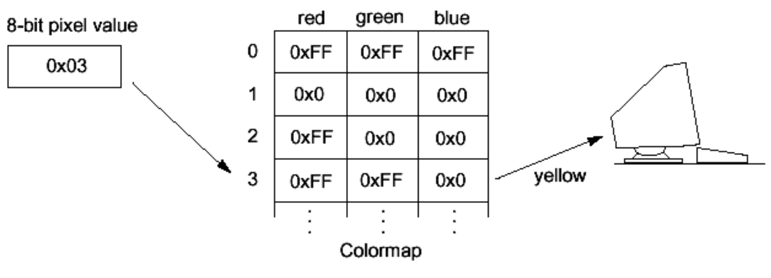
Graphics commands:
- Draw point
- Draw polygon
- Draw text
- Clear frame buffer
- Change drawing color

2 kinds of graphics processors:

2D graphics processors execute commands in 2D coordinates. When objects overlap, the one being drawn will obscure objects drawn previously in the region. **BitBlt** operations (Bit Block Transfer) are usually provided for moving/copying one rectangular region of frame buffer contents to another region.

3D graphics processors execute commands in 3D coordinates. When objects overlap, it is required to determine the visibility of the objects according to the z values.

Display Controller for a raster display device reads the frame buffer and generates the control signals for the screen, ie. the signals for horizontal scanning and vertical scanning. Most display controllers include a **colormap** (or video look-up table). The major function of a colormap is to provide a mapping between the input pixel value to the output color.



	red	green	blue
0	0xFF	0xFF	0xFF
1	0x0	0x0	0x0
2	0xFF	0x0	0x0
3	0xFF	0xFF	0x0
⋮	⋮	⋮	⋮

2.4. Input Devices

Common devices: keyboard, mouse, trackball and joystick

Specialized devices:

Data gloves are electronic gloves for detecting fingers' movement. In some applications, a sensor is also attached to the glove to detect the hand movement as a whole in 3D space.

A **tablet** contains a stylus and a drawing surface and it is mainly used for the input of drawings. A tablet is usually more accurate than a mouse, and is commonly used for large drawings.

Scanners are used to convert drawings or pictures in hardcopy format into digital signal for computer processing.

Touch panels allow displayed objects or screen positions to be selected with the touch of a finger. In these devices a touch-sensing mechanism is fitted over the video monitor screen. Touch input can be recorded using optical, electrical, or acoustical methods.

2.5 Hard-Copy Devices

Directing pictures to a printer or plotter to produce hard-copy output on 35-mm slides, overhead transparencies, or plain paper. The quality of the pictures depend on dot size and number of dots per inch (DPI).

Types of printers: line printers, laserjet, ink-jet, dot-matrix

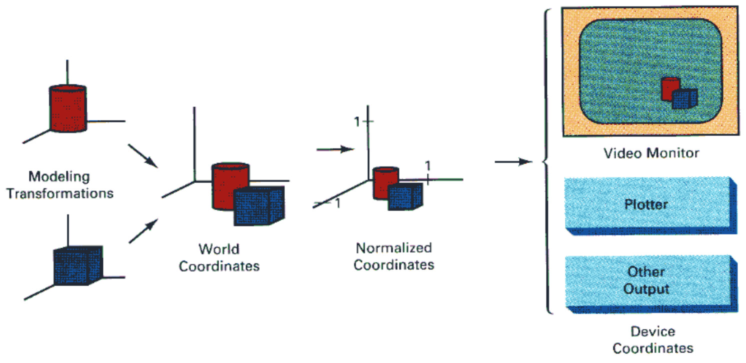
Laserjet printers use a laser beam to create a charge distribution on a rotating drum coated with a photoelectric material. Toner is applied to the drum and then transferred to the paper. To produce color outputs, the 3 color pigments (cyan, magenta, and yellow) are deposited on separate passes.

Inkjet printers produce output by squirting ink in horizontal rows across a roll of paper wrapped on a drum. To produce color outputs, the 3 color pigments are shot simultaneously on a single pass along each print line on the paper.

Inkjet or pen plotters are used to generate drafting layouts and other drawings of normally larger sizes. A pen plotter has one or more pens of different colors and widths mounted on a carriage which spans a sheet of paper.

2.6 Coordinate Representations in Graphics

General graphics packages are designed to be used with Cartesian coordinate representations (x,y,z). Usually several different Cartesian reference frames are used to construct and display a scene:



Modeling coordinates are used to construct individual object shapes.

World coordinates are computed for specifying the placement of individual objects in appropriate positions.

Normalized coordinates are converted from world coordinates, such that x,y values are ranged from 0 to 1.

Device coordinates are the final locations on the output devices.

3. Output Primitives

Shapes and colors of objects can be described internally with pixel arrays or sets of basic geometric structures such as straight line segments and polygon color areas. The functions provided by graphics programming packages to deal with these basic geometric structures are called **output primitives**.

For example:

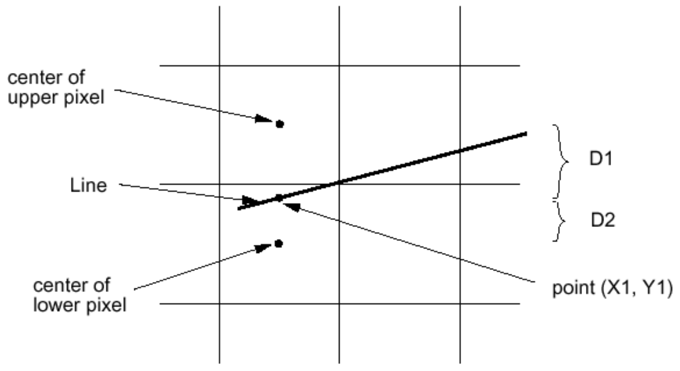
Drawing a point:	SetPixel(100,200,RGB(255,255,0));
Drawing a line:	MoveTo(100,100); LineTo(100,200);
Drawing some text:	SetText(100,200,"Hello");
Drawing an ellipse:	Ellipse(100,100,200,200);
Painting a picture:	BitBlt(100,100,50,50,srcImage,0,0,SRCCOPY);

3.1 Drawing a Thin Line in Raster Devices

This is to compute intermediate discrete coordinates along the line path between 2 specified endpoint positions. The corresponding entry of these discrete coordinates in the frame buffer is then marked with the line color wanted.

The basic concept is:

- A line can be specified in the form:
  $$y = mx + c$$
- Let m be between 0 to 1, then the slope of the line is between 0 and 45 degrees.
- For the x-coordinate of the left end point of the line, compute the corresponding y value according to the line equation. Thus we get the left end point as (x1,y1), where y1 may not be an integer.
- Calculate the distance of (x1,y1) from the center of the pixel immediately above it and call it D1
- Calculate the distance of (x1,y1) from the center of the pixel immediately below it and call it D2
- If D1 is smaller than D2, it means that the line is closer to the upper pixel than the lower pixel, then, we set the upper pixel to on; otherwise we set the lower pixel to on.
- Then increatement x by 1 and repeat the same process until x reaches the right end point of the line.
- This method assumes the width of the line to be zero



Bresenham's Line Algorithm

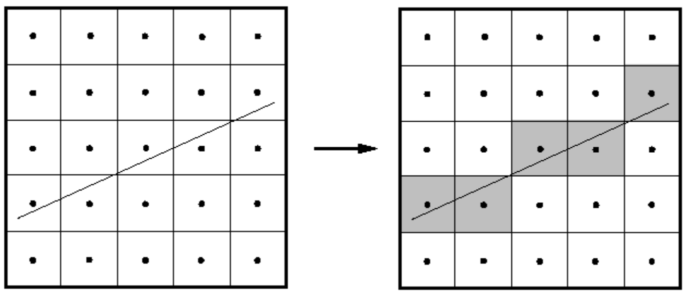
This algorithm is very efficient since it use only incremental integer calculations. Instead of calculating the non-integral values of D1 and D2 for decision of pixel location, it computes a value, p, which is defined as:

p = (D2-D1)* horizontal length of the line
if p>0, it means D1 is smaller than D2, and we can determine the pixel location accordingly

However, the computation of p is very easy:
The initial value of p is 2 * vertical height of the line - horizontal length of the line.
At succeeding x locations, if p has been smaller than 0, then, we increment p by 2 * vertical height of the line, otherwise we increment p by 2 * (vertical height of the line - horizontal length of the line)

All the computations are on integers. The incremental method is applied to

```
void BresenhamLine(int x1, int y1, int x2, int y2)
{  int x, y, p, const1, const2;
   /* initialize variables */
   p=2*(y2-y1)-(x2-x1);
   const1=2*(y2-y1);
   const2=2*((y2-y1)-(x2-x1));

   x=x1;
   y=y1;
   SetPixel(x,y);
   while (x<xend)
   {  x++;
      if (p<0)
      {  p=p+const1;
      }
      else
      {  y++;
         p=p+const2;
      }
      SetPixel(x,y);
   }
}
```



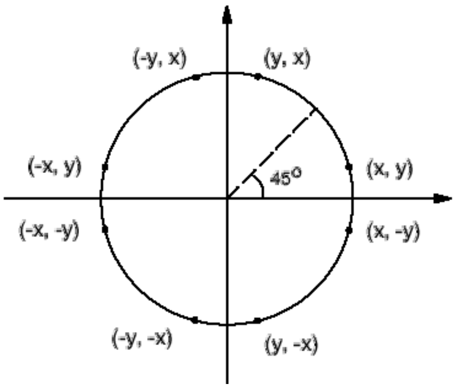
3.2 Drawing a Circle in Raster Devices

A circle can be specified in the form:

$$(x-x_c)^2 + (y-y_c)^2 = r^2$$

where (x_c,y_c) is the center of the circle.

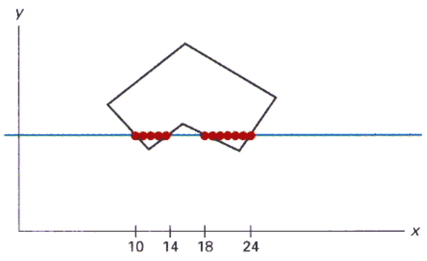
To save time in drawing a circle, we can make use of the symmetrical property of a circle which is to draw the segment of the circle between 0 and 45 degrees and repeat the segment 8 times as shown in the diagram to produce a circle. Ths algorithm also employs the incremental method which further improves the efficiency.



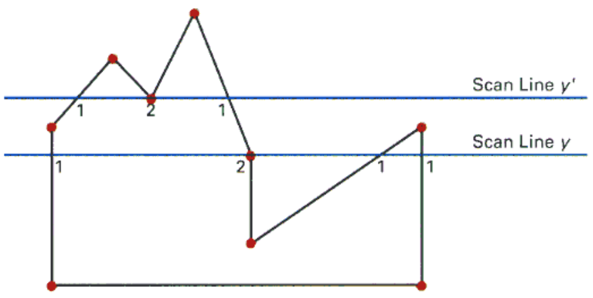
```
void PlotCirclePoints(int centerx, int centery, int x, int y)
{  SetPixel(centerx+x,centery+y);
   SetPixel(centerx-x,centery+y);
   SetPixel(centerx+x,centery-y);
   SetPixel(centerx-x,centery-y);
   SetPixel(centerx+y,centery+x);
   SetPixel(centerx-y,centery+x);
   SetPixel(centerx+y,centery-x);
   SetPixel(centerx-y,centery-x);
}

void BresenhamCircle(int centerx, int centery, int radius)
{  int x=0;
   int y=radius;
   int p=3-2*radius;
   while (x<y)
   {  PlotCirclePoints(centerx,centery,x,y);
      if (p<0)
      {  p=p+4*x+6;
      }
      else
      {  p=p+4*(x-y)+10;
         y=y-1;
      }
      x=x+1;
   }
   PlotCirclePoints(centerx,centery,x,y);
}
```

3.3 Scan-Line Polygon Fill Algorithm

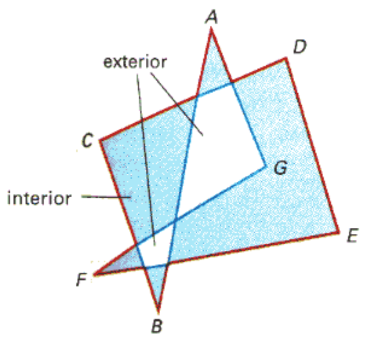
- Basic idea: For each scan line crossing a polygon, this algorithm locates the intersection points of the scan line with the polygon edges. These intersection points are shorted from left to right. Then, we fill the pixels between each intersection pair.



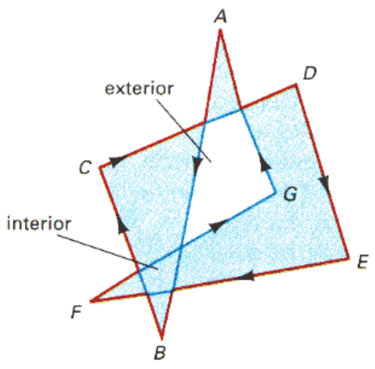
- Some scan-line intersection at polygon vertices require special handling. A scan line passing through a vertex as intersecting the polygon twice. In this case we may or may not add 2 points to the list of intersections, instead of adding 1 points. This decision depends on whether the 2 edges at both sides of the vertex are both above, both below, or one is above and one is below the scan line. Only for the case if both are above or both are below the scan line, then we will add 2 points.



- **Inside-Outside Tests:** The above algorithm only works for standard polygon shapes. However, for the cases which the edges of the polygon intersects, we need to identify whether a point is an interior or exterior point. Students may find interesting descriptions of 2 methods to solve this problem in many text books: odd-even rule and nonzero winding number rule.



Odd Even rule

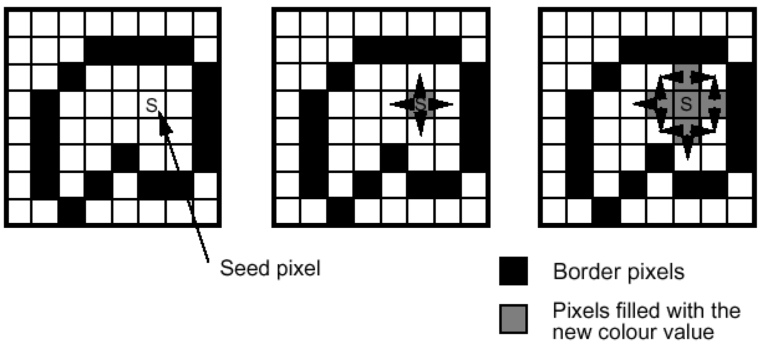


Non. ero Winding Number Rule

3.4 Boundary-Fill Algorithm

- This algorithm starts at a point inside a region and paint the interior outward towards the boundary.
- This is a simple method but not efficient: 1. It is recursive method which may occupy a large stack size in the main memory.

```
void BoundaryFill(int x, int y, COLOR fill, COLOR boundary)
{  COLOR current;
   current=GetPixel(x,y);
   if (current<>boundary) and (current<>fill) then
   {  SetPixel(x,y,fill);
      BoundaryFill(x+1,y,fill,boundary);
      BoundaryFill(x-1,y,fill,boundary);
      BoundaryFill(x,y+1,fill,boundary);
      BoundaryFill(x,y-1,fill,boundary);
   }
}
```



- More efficient methods fill horizontal pixel spands across scan lines, instead of proceeding to neighboring points.

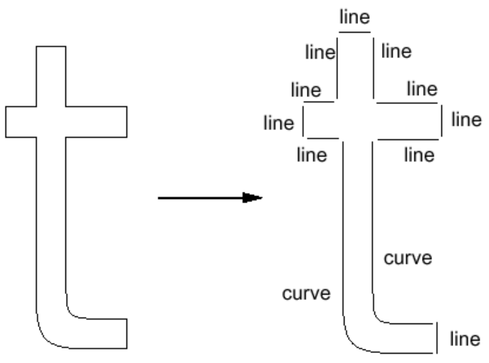
3.5 Flood-Fill Algorithm

- Flood-Fill is similar to Boundary-Fill. The difference is that Flood-Fill is to fill an area which I not defined by a single boundary color.

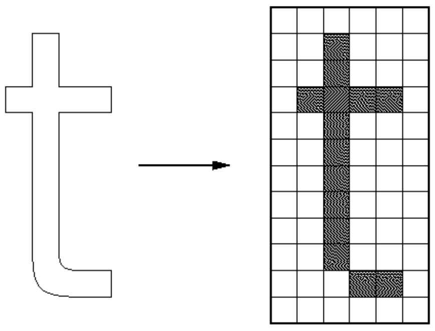
```
void BoundaryFill(int x, int y, COLOR fill, COLOR old_color)
{  if (GetPixel(x,y)== old_color)
   {  SetPixel(x,y,fill);
      BoundaryFill(x+1,y,fill,boundary);
      BoundaryFill(x-1,y,fill,boundary);
      BoundaryFill(x,y+1,fill,boundary);
      BoundaryFill(x,y-1,fill,boundary);
   }
}
```

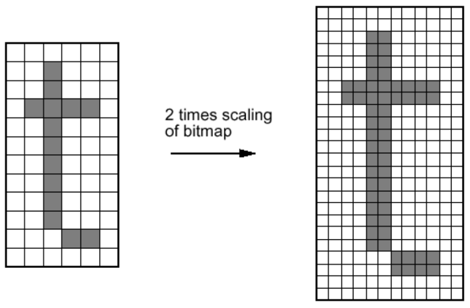
3.6 Drawing Text

- A character is defined by its outline which is usually comoposed of lines and curves.



- We can use a method similar the one for rendering polygon to render a character
- However, because text is used very oftenly, we usually convert them into bitmaps in advance to improve the drawing efficiency.
- To draw a character on the screen, all we need to do is to copy the corresponding bitmap to the specified coordinate.
- The problem with this method is that scaling a character with a bitmap to produce different character sizes would result in a block-like structures (stair-case, aliasing). Hence we normally render a few bitmaps for a single character to represent different sizes of the same character.





3.7 Bitmap

- A graphics pattern suh as an icon or a character may be needed frequently, or may need to be re-used.
- Generating the pattern every time when needed may waste a lot of processing time.
- A bitmap can be used to store a pattern and duplicate it to many places on the image or on the screen with simple copying operations.

3.8 Properties

- In graphical output primitives, objects are normally associated with properties. Eg.

Point:	color
Line:	width, style, color
Polygon:	edge color, filling color
Text:	font size, color, bold or not bold, italic or not, underlined or not, etc.

In graphical packages, we can specify such properties, eg. In Powerpoint, we can modify the properties of objects by a format command.

In programming tools, we may pass the properties as arguments when we call the functions of these primitives, or we may pre-select the properties before calling the functions.