Unit 1: Graphics Primitives

Introduction

The term computer graphics includes almost everything on computers that is not text or sound. Today almost every computer can do some graphics, and people have even come to expect to control their computer through icons and pictures rather than just by typing. Here in our lab at the Program of Computer Graphics, we think of computer graphics as drawing pictures on computers, also called rendering. The pictures can be photographs, drawings, movies, or simulations - pictures of things, which do not yet exist and maybe could never exist. Or they may be pictures from places we cannot see directly, such as medical images from inside your body. We spend much of our time improving the way computer pictures can simulate real world scenes. We want images on computers to not just look more realistic, but also to be more realistic in their colors, the way objects and rooms are lighted, and the way different materials appear. We call this work “realistic image synthesis”.

Interactive Graphics

In interactive computer graphics user have some control over the picture i.e user can make any change in the produced image. One example of it is the ping pong game. The conceptual model of any interactive graphics system is given in the picture shown in Figure 1.1. At the hardware level (not shown in picture), a computer receives input from interaction devices, and outputs images to a display device. The software has three components. The first is the application program, it creates, stores into, and retrieves from the second component, the application model, which represents the the graphic primitive to be shown on the screen. The application program also handles user input. It produces views by sending to the third component, the graphics system, a series of graphics output commands that contain both a detailed geometric description of what is to be viewed and the attributes describing how the objects should appear. After the user input is processed, it sent to the graphics system is for actually producing the picture. Thus the graphics system is a layer in between the application program and the display hardware that effects an output transformation from objects in the application model to a view of the model.

1.3 Passive Graphics

A computer graphics operation that transfers automatically and without operator intervention. Non-interactive computer graphics involves one way communication between the computer and the user. Picture is produced on the monitor and the user does not have any control over the produced picture.

1.4 Advantages of Interactive Graphics

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In Many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the late 1980s, when scientists and engineers realized that they could not interpret the data and prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

Creating and reproducing pictures, however, presented technical problems that stood in the way of their widespread use. Thus, the ancient Chinese proverb "a picture is worth ten thousand words" became a cliché in our society only after the advent of

inexpensive and simple technology for producing pictures—first the printing press, then photography.

Interactive computer graphics is the most important means of producing pictures since the invention of photography and television; it has the added advantage that, with the computer, we can make pictures not only of concrete, "real-world" objects but also of abstract, synthetic objects, such as mathematical surfaces in 4D and of data that have no inherent geometry, such as survey results. Furthermore, we are not confined to static images. Although static pictures are a good means of communicating information, dynamically varying pictures are frequently even better–to time-varying phenomena, both real (e.g., growth trends, such as nuclear energy use in the United States or population movement form cities to suburbs and back to the cities). Thus, a movie can show changes over time more graphically than can a sequence of slides. Thus, a sequence of frames displayed on a screen at more than 15 frames per second can convey smooth motion or changing form better than can a jerky sequence, with several seconds between individual frames. The use of dynamics is especially effective when the user can control the animation by adjusting the speed, the portion of the total scene in view, the amount of detail shown, the geometric relationship of the objects in the another, and so on. Much of interactive graphics technology therefore contains hardware and software for user-controlled motion dynamics and update dynamics.

With motion dynamics, objects can be moved and tumbled with respect to a stationary observer. The objects can also remain stationary and the viewer can move around them , pan to select the portion in view, and zoom in or out for more or less detail, as though looking through the viewfinder of a rapidly moving video camera. In many cases, both the objects and the camera are moving. A typical example is the flight simulator, which combines a mechanical platform supporting a mock cockpit with display screens for windows. Computers control platform motion, gauges, and the simulated world of both stationary and moving objects through which the pilot navigates. These multimillion-dollar systems train pilots by letting the pilots maneuver a simulated craft over a simulated 3D landscape and around simulated vehicles. Much simpler fight simulators are among the most popular games on personal computers and workstations.

Amusement parks also offer “motionsimulator” rides through simulated terrestrial and extraterrestrial landscapes. Video

arcades offer graphics-based dexterity games and racecar-driving simulators, video games exploiting interactive motion dynamics: The player can change speed and direction with the “gas pedal” and “steering wheel,” as trees, buildings, and other cars go whizzing by. Similarly, motion dynamics lets the user fly around the through buildings, molecules, and 3D or 4D mathematical space. In another type of motion dynamics, the “camera” is held fixed, and the objects in the scene are moved relative to it. For example, a complex mechanical linkage, such as the linkage on a stream engine, can be animated by moving or rotating all the pieces appropriately.

Update dynamics is the actual change of the shape, color, or other properties of the objects being viewed. For instance, a system can display the deformations of an airplane structure in flight or the state changes in a block diagram of a nuclear reactor in response to the operator’s manipulation of graphical representations of the many control mechanisms. The smoother the change, the more realistic and meaningful the result. Dynamic interactive graphics offers a large number of user-controllable modes with which to encode and communicate information: the 2D or 3D shape of objects in a picture, their gray scale or color, and the time variations of these properties. With the recent development of digital signal processing (DSP) and audio synthesis chips, audio feedback can now be provided to augment the graphical feedback and to make the simulated environment even more realistic.

Interactive computer graphics thus permits extensive, high-bandwidth user-computer interaction. This significantly enhances our ability to understand data, to perceive trends, and to visualize real or imaginary objects–indeed, to create “virtual worlds” that we can explore from arbitrary points of view. By making communication more efficient, graphics make possible higher-quality and more precise results or products, greater productivity, and lower analysis and design costs.

1.6 Applications of Computer Graphics Classification of Applications The diverse uses of computer graphics listed in the previous section differ in a variety of ways, and a number of classification is by type (dimensionality) of the object to be represented and the kind of picture to be produced

Cartography: Computer graphics is used to produce both accurate and schematic representations of geographical and other natural phenomena from measurement data. Examples include geographic maps, relief maps, exploration maps for drilling and mining, oceanographic charts, weather maps, contour maps, and population-density maps.

User interfaces: As soon mentioned, most applications that run on personal computers and workstations, and even those that run on terminals attached to time shared computers and network compute servers, have user interfaces that rely on desktop window systems to manage multiple simultaneous activities, and on point and click facilities to allow users to select menu items, icons, and objects on the screen; typing is necessary only to input text to be stored and manipulated. Word-processing , spreadsheet, and desktop-publishing programs are typical applications that take

Advantage of such user-interface techniques: The authors of this book used such programs to create both the text and the figures; then , the publisher and their contractors produced the book using similar typesetting and drawing software.

(Interactive) plotting in business, science and technology: The next most common use of graphics today is probably to create 2D and 3D graphs of mathematical, physical, and economic functions; histograms, bar and pie charts; task-scheduling charts; inventory and production charts, and the like . All these are used to present meaningfully and concisely the trends and patterns gleaned from data, so as to clarify complex phenomena and to facilitate informed decision making.

Office automation and electronic publishing: The use of graphics for the creation and dissemination of information has increased enormously since the advent of desktop publishing on personal computers. Many organizations whose publications used to be printed by outside specialists can now produce printed materials inhouse. Office automation and electronic publishing can produce both traditional printed (hardcopy) documents and electronic (softcopy) documents that allow browsing of networks of interlinked multimedia documents are proliferating

Computer-aided drafting and design: In computer-aided design (CAD), interactive graphics is used to design components and systems of mechanical , electrical, electromechanical, and electronic devices, including structure such as buildings, automobile bodies, airplane and ship hulls, very large scale-integrated (VLSI) chips, optical systems, and telephone and computer networks. Sometimes, the use; merely wants to produce the precise drawings of components and assemblies, as for online drafting or architectural blueprints Color Plate 1.8 shows an example of such a 3D design program, intended for nonprofessionals also a customize your own patio deck" program used in lumber yards. More frequently however the emphasis is on interacting with a computer based model of the component or system being designed in order to test, for example, its structural, electrical, or thermal properties. Often, the model is interpreted by a simulator that feeds back the behavior of the system to the user for further interactive design and test cycles. After objects have been designed,

utility programs can postprocess the design database to make parts lists, to process 'bills of materials', to define numerical control tapes for cutting or drilling parts, and so on.

Simulation and animation for scientific visualization and entertainment: Computer produced animated movies and displays or the time-varying behavior of real and simulated objects are becoming increasingly popular for scientific and engineering visualization. We can use them to study abstract mathematical entries as well as mathematical models of such phenomena as fluid flow, relativity, nuclear and chemical reactions, physiological system and organ function, and deformation of mechanical structures under various kinds of loads. Another advanced-technology area is interactive cartooning. The simpler kinds of systems for producing ‘Flat” cartons are becoming cost-effective in creating routine ‘in-between” frames that interpolate between two explicity specified ‘key frames”. Cartoon characters will increasingly be modeled in the computer as 3D shape descriptions whose movements are controlled by computer commands, rather than by the figures being drawn manually by cartoonists . Television commercials featuring flying logos and more exotic visual trickery have become common, as have elegant special effects in movies. Sophisticated mechanisms are available to model the objects and to represent light and shadows.

Art and commerce: Overlapping the previous categories the use of computer graphics in art and advertising here, computer graphics is used to produce pictures that express a message and attract attention. Personal computers and Teletext and Videotexts terminals in public places such as in private homes, offer much simpler but still informative pictures that let users orient themselves, make choices, or even “teleshop” and conduct other business transactions. Finally, slide production for commercial, scientific, or educational presentations is another cost-effective use of graphics, given the steeply rising labor costs of the traditional means of creating such material.

Process control: Whereas flight simulators or arcade games let users interact with a simulation of a real or artificial world, many other applications enable people or interact with some aspect of the real world itself. Status displays in refineries, power plants, and computer networks show data values from sensors attached to critical system components, so that operators can respond to problematic conditions. For example, military commanders view field data – number and position of vehicles, weapons launched, troop movements, causalities – on command and control displays to revise their tactics as needed; flight controller airports see computer-generated identification and status information for the aircraft blips on their radar scopes, and can thus control traffic more quickly and accurately than they could with the uninitiated radar data alone; spacecraft controllers monitor telemetry data and take corrective action as needed.

Display Devices

The principle of producing images as collections of discrete points set to appropriate colours is now widespread throughout all fields of image production. The most common graphics output device is the video monitor which is based on the standard cathode ray tube(CRT) design, but several other technologies exist and solid state monitors may eventually predominate.

Refresh CRT

Basic Operation of CRT

A beam of electrons (cathode rays), emitted by an electron gun, passes through focusing and deflection systems that direct the beam toward specified positions on the phosphor-coated screen. The phosphor then emits a small spot of light at each position contacted by the electron beam. Because the light emitted by the phosphor fades very rapidly, some method is needed for maintaining the screen picture. One Way to keep the phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back over the same points. This type of display is called a refresh CRT.

Working

Beam passes between two pairs of metal plates, one vertical and other horizontal. A voltage difference is applied to each pair of plates according to the amount that the beam is to be deflected in each direction. As the electron beam passes between each pair of plates, it is bent towards the plate with the higher positive voltage. In figure

2.2 the beam is first deflected towards one side of the screen. Then, as the beam passes through the horizontal plates, it is deflected towards, the top or bottom of the screen. To get the proper deflection, adjust the current through coils placed around the outside of the CRT loop. The primary components of an electron gun in a CRT are the heated metal cathode and a control grid (Fig. 2.2). Heat is supplied to the cathode by directing a current through a coil of wire, called the filament, inside the cylindrical cathode structure. This causes electrons to be "boiled off" the hot cathode surface. In the vacuum inside the CRT envelope, the free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage. The accelerating voltage can be generated with a positively charged metal coating on the in- side of the CRT envelope near the phosphor screen, or an accelerating anode can be used.

Random-Scan and Raster Scan Monitor

Random-Scan/Calligraphic displays

Random scan system uses an electron beam which operates like a pencil to create a line image on the CRT. The image is constructed out of a sequence of straight line segments. Each line segment is drawn on the screen by directing the beam to move from one point on screen to the next, where each point is defined by its x and y coordinates. After drawing the picture, the system cycles back to the first line and design all the lines of the picture 30 to 60 time each second. When operated as a random-scan display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn. Random-scan monitors draw a picture one line at a time and for this reason are also referred to as vector displays (or strokewriting or calligraphic displays). A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device. Refresh rate on a random-scan system depends on the number of lines to be displayed. Picture definition is now stored as a set of line-drawing commands in an area of memory referred to as the refresh display file. Random-scan systems are designed for line-drawing applications and can-not display realistic shaded scenes. Since picture definition is stored as a set of line-drawing instructions and not as a set of intensity values for all screen points, vector displays generally have higher resolution than raster systems. Also, vector displays produce smooth line drawings because the CRT beam directly follows the line path

Raster-Scan Displays

In raster scan approach, the viewing screen is divided into a large number of discrete phosphor picture elements, called pixels. The matrix of pixels constitutes the raster. The number of separate pixels in the raster display might typically range from 256X256 to 1024X 1024. Each pixel on the screen can be made to glow with a different brightness. Colour screen provide for the pixels to have different colours as well as brightness. In a raster-scan system, the electron beam is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. Picture definition is stored in a memory area called the refresh buffer or frame buffer. This memory area holds the set of intensity values for all the screen points. Stored intensity values are then retrieved from the refresh buffer and "painted" on the screen one row (scan line) at a time. Each screen point is referred to as a pixel or pel (shortened forms of picture element). The capability of a raster-scan system to store intensity information for each screen point makes it well suited for the realistic display of scenes containing subtle shading and color patterns. Home television sets and printers are examples of other systems using raster-scan methods.

Intensity range for pixel positions depends on the capability of the raster system. In a simple black-and-white system, each screen point is either on or off, so only one bit per pixel is needed to control the intensity of screen positions. For a bilevel system, a bit value of 1 indicates that the electron beam is to be turned on at that position, and a value of 0 indicates that the beam intensity is to be off. Additional bits are needed when color and intensity variations can be displayed. On some raster-scan systems (and in TV sets), each frame is displayed in two passes using an interlaced refresh procedure. In the first pass, the beam sweeps across every other scan line from top to bottom. Then after the vertical re- trace, the beam sweeps out the remaining scan lines (Fig. 2.7). Interlacing of the scan lines in this way allows us to see the entire screen displayed in one-half the time it would have taken to sweep across all the lines at once from top to bottom. Interlacing is primarily used with slower refreshing rates. On an older, 30 frame- per-second, noninterlaced display, for instance, some flicker is noticeable. But with interlacing, each of the two passes can be accomplished in l/60th of a second, which brings the refresh rate nearer to 60 frames per second. This is an effective technique for avoiding flicker, providing that adjacent scan lines contain similar display information.

Scan Converting Line

Introduction

We have studied various display devices in the previous chapter. It is clear that these devices need special procedures for displaying any graphic object: line, circle, curves, and even characters. Irrespective of the procedures used, the system can generate the images on these raster devices by turning the pixels on or off. The process in which the object is represented as the collection of discrete pixels is called scan conversion. The video output circuitry of a computer is capable of converting binary values stored in its display memory into pixel-on, pixel-off information that can be used by a raster output device to display a point. This ability allows graphics computers to display models composed of discrete dots.

Almost any model can be reproduced with a sufficiently dense matrix of dots (pointillism), most human operators generally think in terms of more complex graphics objects such as points, lines, circles and ellipses. Since the inception of computer graphics, many algorithms have been developed to provide human users with fast, memory-efficient routines that generate higher-level objects of this kind. However, regardless of what routines are developed, the computer can produce images on raster devices only by turning the appropriate pixels on or off. Many scanconversion algorithms are implemented in computer hardware or firmware. However, a specific graphics algorithm, the scan-conversion algorithm can be implemented in software. The most commonly used graphics objects are the line, the sector, the arc, the ellipse, the rectangle and the polygon.

3.2 Scan-converting a Point

We have already defined that a pixel is collection of number of points. Thus it does not represent any mathematical point. Suppose we wish to display a point C(5.4, 6.5). It means that we wish to illuminate that pixel, which contains this point C. Refer to figure 3.1, which shows that pixel corresponding to point C. What happens if we try to display C'(5.3, 6.4)? Well, it also corresponding to the same pixel as that of C (5.4,6.5).Thus we can say that point C(x,y) is represented by an integer part of x and integer part of y. So, we can use the command as

Putpixel(int x, int y);

We normally use right handed cartesian coordinate system. The origin in this system starts at the bottom. However in case of computer system, due to the memory organization, the system turns out ot left handed Cartesian system. Thus there is a difference in the actual representation and the way in which we work with the points. The basic steps involved in converting Cartesian coordinate system to the system understable points are: Step 1: Identify the starting address corresponding to the line on which the point is to be displayed. Step 2: Find the byte address in which the point is to be displayed. Step 3: Compute the value for the byte that represents the point. Step 4: Logically OR the calculated value with the present value of the byte. Step 5: Store the value found in step 4 in the byte found in steps 1 and 2. Step 6: Stop. 3.3 Scan-converting a Straight Line

A scan conversion of line locates the coordinates of the pixels lie on or near an ideal straight line impaired on 2D raster grid. Before discussing the various methods, let us see what are the characteristics of line. One expects the following features of line:

1. The line should appear straight line.

2. The line should have equal brightness throughout their length.

3. The line must be drawn rapidly.

Even though the rasterization tries to generate a completely straight line, yet in few cases we may not get equal brightness. Basically, the lines which are horizontal or vertical or oriented by 450 , have equal brightness. But for the lines with larger length and different orientations, we need to have complex computations in our algorithms. This may reduce the speed of generation of line. Thus we make some sort of compromise while generating the lines, such as:

1. Calculate only the approximate line length.

2. Make use of simple arithmetic computations, preferabley integer arithmetic.

3. Implement result in hardware or firmware.

A straight line may be defined by two endpoints and an equation figure 3.2. In figure 3.1 the two endpoints are described by (x1, y1) and (x2, y2). The equation of the line is used to describe the x, y coordinates of all the points that lie between these two endpoints. Using the equation of a straight line, y = mx + b where m = Δy/Δx and b = the y intercept, we can find values of y by incrementing x = x1 to x = x2. By scan converting these calculated x = x2. By scan-converting these calculated x, y values, we represent the line as a sequence on pixels.

While this method of scan-converting a straight line is adequate for many graphics applications, interactive graphics systems require a much faster response than the method described above can provide. Interactive graphics is a graphics system in which the user dynamically controls the presentation of graphics models on a computer display.

Direct Use of Line Equation

A simple approach to scan-converting a line is to first scan-convert P1 and P2 to pixel coordinates (x'1, y'1) and (x'2, y'2), respectively; then set m = (y'2, y'1) (x'2, x'1) and b = y'1 –mx'1. If [m] then for every integer value of x between and excluding x' 1,≤ 1 and x'2, calculate the corresponding value of y using the equation and scan-convert (x, y). If [m] > 1, then for every integer value of y between and excluding y'1 and y'2 calculate the corresponding value of x using the equation and scan-convert (x, y).

While this approach is mathematically sound, it involves floating-point computation (multiplication and addition) in every step that uses the line equation since m and b are generally real numbers. The challenge is to find a way to achieve the same goal as quickly as possible.

3.3.2 DDA Algorithm This algorithm works on the principle of obtaining the successive pixel values basede on the differential equation goverening the line. Since screen pixels are referred with integer values, or plotted positions, which may only approximate the calculated coordinates – i.e., pixels which are intensified are those which lie very close to the line path if not exactly on the line path which in this case are perfectly horizontal, vertical or 45° lines only. Standard algorithms are available to determine which pixels provide the best approximation to the desired line, one such algorithm is the DDA (Digital Differential Analyser) algorithm. Before going to the details of the algorithm, let us discuss some general appearances of the line segment, because the respective appearance decides which pixels are to be intensified. It is also obvious that only those pixels that lie very close to the line path are to be intensified because they are the ones which best approximate the line. Apart from the exact situation of the line path, which in this case are perfectly horizontal, vertical or 45° lines (i.e., slope zero, infinite, one) only. We may also face a situation where the slope of the line is > 1 or < 1.Which is the case shown in Figure 3.3.

In Figure 3.3, there are two lines. Line 1 (slope<1) and line 2 (slope>1). Now let us discuss the general mechanism of construction of these two lines with the DDA algorithm. As the slope of the line is a crucial factor in its construction, let us consider the algorithm in two cases depending on the slope of the line whether it is > 1 or < 1. Case 1: slope (m) of line is < 1 (i.e., line 1): In this case to plot the line we have to move the direction of pixel in x by 1 unit every time and then hunt for the pixel value of the y direction which best suits the line and lighten that pixel in order to plot the line. So, in Case 1 i.e., 0 < m < 1 where x is to be increased then by 1 unit every time and proper y is approximated.

Case 2: slope (m) of line is > 1 (i.e., line 2) if m > 1 i.e., case of line 2, then the most appropriate strategy would be to move towards the y direction by 1 unit every time and determine the pixel in x direction which best suits the line and get that pixel lightened to plot the line. So, in Case 2, i.e., (infinity) > m > 1 where y is to be increased by 1 unit every time and proper x is approximated.

<u>**Bresenham’s Line Algorithm**</u>

Bresenham's line-drawing algorithm uses an iterative scheme. A pixel is plotted at the starting coordinate of the line, and each iteration of the algorithm increments the pixel one unit along the major, or x-axis. The pixel is incremented along the minor, or yaxis, only when a decision variable (based on the slope of the line) changes sign. A key feature of the algorithm is that it requires only integer data and simple arithmetic. This makes the algorithm very efficient and fast.

The algorithm assumes the line has positive slope less than one, but a simple change of variables can modify the algorithm for any slope value.

Bresenham's Algorithm for 0 < slope < 1 Figure 3.4 shows a line segment superimposed on a raster grid with horizontal axis X and vertical axis Y. Note that xi and yi are the integer abscissa and ordinate respectively of each pixel location on the grid. Given (xi, yi) as the previously plotted pixel location for the line segment, the next pixel to be plotted is either (xi + 1, yi) or (xi + 1, yi + 1). Bresenham's algorithm determines which of these two pixel locations is nearer to the actual line by calculating the distance from each pixel to the line, and plotting that pixel with the smaller distance. Using the familiar equation of a straight line, y = mx + b, the y value corresponding to xi + 1 is y=m(xi+1) + b The two distances are then calculated as: d1 = y- yi d1= m(xi + 1) + b- yi d2 = (yi+ 1)- y d2 = (yi+ 1) - m(xi + 1)- b and, d1 - d2 = m(xi+ 1) + b - yi - (yi + 1) + m(xi + 1) + b d1 - d2 = 2m(xi + 1) - 2yi + 2b - 1 Multiplying this result by the constant dx, defined by the slope of the line m = dy/dx, the equation becomes: dx(d1-d2)= 2dy(xi)- 2dx(yi) + c

where c is the constant 2dy + 2dxb - dx. Of course, if d2 > d1, then (d1-d2) < 0, or conversely if d1 > d2, then (d1-d2) >0. Therefore, a parameter pi can be defined such that pi = dx(d1-d2)

pi = 2dy(xi) - 2dx(yi) + c If pi > 0, then d1 > d2 and yi + 1 is chosen such that the next plotted pixel is (xi + 1, yi). Otherwise, if pi < 0, then d2 > d1 and (xi + 1, yi + 1) is plotted. (See Figure3.5 .) Similarly, for the next iteration, pi + 1 can be calculated and compared with zero to determine the next pixel to plot. If pi +1 < 0, then the next plotted pixel is at (xi + 1 + 1, Yi+1); if pi + 1< 0, then the next point is (xi + 1 + 1, yi + 1 + 1). Note that in the equation for pi + 1, xi + 1 = xi + 1. pi + 1 = 2dy(xi + 1) - 2dx(yi + 1) + c. Subtracting pi from pi + 1, we get the recursive equation: pi + 1 = pi + 2dy - 2dx(yi + 1 - yi) Note that the constant c has conveniently dropped out of the formula. And, if pi < 0 then yi + 1= yi in the above equation, so that: pi + 1 = pi + 2dy or, if pi > 0 then yi + 1 = yi + 1, and pi + 1 = pi + 2(dy-dx) To further simplify the iterative algorithm, constants c1 and c2 can be initialized at the beginning of the program such that c1 = 2dy and c2 = 2(dy-dx). Thus, the actual meat of the algorithm is a loop of length dx, containing only a few integer additions and two compares (Figure 3.5) .

Scan-converting a Circle

Circle is one of the basic graphic component, so in order to understand its generation, let us go through its properties first. A circle is a symmetrical figure. Any circlegenerating algorithm can take advantage of the circle’s symmetry to plot of eight points for each value that the algorithm calculates. Eight-way symmetry is used by reflecting each calculated point around each 45˚ axis. For example, if point 1 in Fig. 3.6 were calculated with a circle algorithm, seven more points could be found by reflection. The reflection is accomplished by reversing the x, y coordinates as in point 2, reversing the x,y coordinates and reflecting about the y axis as in point 3, reflecting about the y axis in point 4, switching the signs of x and y as in point 5, reversing the x, y coordinates, reflecting about the y axis and reflecting about the x axis.

As in point 6, reversing the x, y coordinates and reflecting about the y-axis as in point 7, and reflecting about the x-axis as in point 8.

To summarize:

P 1 = (x, y) P5 = (- y,-x)

P 2 = (y, x) P6 = (-y, - x)

P 3 = (-y, x) P7 = (y, -x)

P 4 = (-x, y) P8 = (x, -y)

Defining a Circle

There are two standard methods of mathematically defining a circle centered at the origin. The first method defines a circle with the second-order polynomial equation (see Fig. 3.7). y2 = r2 – x2

Where x = the x coordinate

y = the y coordinate

r = the circle radius

With this method, each x coordinate in the sector, from 90 to 45°, is found by stepping x from 0 to r/ 2, and each y coordinate is found by evaluating 2 2 r x + for each step of x. This is a very inefficient method, however, because for each point both x and r must be squared and subtracted from each other; then the square root of the result must be found.

The second method of defining a circle makes use of trigonometric functions (see Fig. 3.8):

x = r cos θ y = r sin θ

where θ = current angle

r = circle radius

x = x coordinate

y = y coordinate

By this method, θ is stepped from θ to π/4, and each value of x and y is calculated. However, computation of the values of sin θ and cos θ is even more time-consuming than the calculations required by the first method.

Bresenham’s Circle Algorithm

If a circle is to be plotted efficiently, the use of trigonometric and power functions must be avoided. And as with the generation of a straight line, it is also desirable to perform the calculations necessary to find the scan-converted points with only integer addition, subtraction, and multiplication by powers of 2. Bresenham’s circle algorithm allows these goals to be met.

Scan-converting a circle Bresenham’s algorithm works as follows. If the eight-way symmetry of a circle is used to generate a circle, points will only have to be generated through a 45°, moves will be made only in the +x and –y directions

The best approximation of the true circle will be described by those pixels in the raster that fall the least distance from the true circle. Notice that if points are generated from 90 and 45°, each new point closest to the true circle can be found by taking either of two actions:(1) move in the x direction one unit or (2) move in the x direction one unit. Therefore, a method of selecting between these two choices is all that is necessary to find the points closets to the true circle.

Mid point Circle Algorithm

We present another incremental circle algorithm that is very similar to Bresenham’s approach. It is based on the following function for testing, the spatial relationship between an arbitrary (x, y) and a circle of radius r centered at the origin:

<0 (X,Y) inside the circle
=0 (X,Y) on the circle

>0 (X,Y) outside the circle
Now consider the coordinates of the point halfway between pixel T and pixel S $(x_i + 1, y_i –).21$

This is called the midpoint and we use it to define decision parameter:

$p_i = f (x_i + 1, y_i –)21= (x_i + 1)^2 + (y_i –)221 – r^2$

If p_i is negative, the midpoint is inside the circle, and we choose pixel T. On the other hand, if p_i is positive (or equal to zero), the midpoint is outside the circle (or on the circle), and we choose pixel S. Similarly, the decision parameter for the next step is

$p_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1 –})221– r^2$
Since $x_{i+1} = x_i + 1$, we have

Since $x_{i+1} = x_i + 1$, we have

Finally, we compute the initial value for the decision parameter using the original definition of p_i and (0, r):

The following is a description of this midpoint circle algorithm that generates the pixel coordinates in the 90^o to 45^o octant:

int x = 0, y = r, p = 1 – r;

```
while (x < = y) {
setPixel (x, y);
If (p < 0)
p = p + 2x + 3;
else {
p = p + 2 (x – y) + 5;
y –;
}
x++;
```

Scan-converting an Ellipse

The ellipse, like the circle, shows symmetry. In the case of an ellipse, however, symmetry is four-rather than eight-way. There are two methods of mathematically defining an ellipse

Steps for generating ellipse using polynomial method are:

1. Set the initial variables: a = length of majo0r axis; b = length of minor axis; (h, k) = coordinates of ellipse center; x = 0; i = step; x_{end} = a.

2. Test to determine whether the entire ellipse has been scan-converted. If x> x_{end}, stop.

3. Compute the value of the y coordinate:

4. Plot the four points, found by symmetry, at the current (x, y) coordinates:

Plot (x + h, y + k) Plot (-x + h, -y + k)

Plot (-y - h, x + k) Plot (y + h, -x + k)

5. Increment x; x = x + i.
6. Go to step 2.

Unit 2: Two Dimensional Transformations

Transformations are fundamental part of computer graphics. In order to manipulate object in two dimensional space, we must apply various transformation functions to object. This allows us to change the position, size, and orientation of the objects. Transformations are used to position objects, to shape objects, to change viewing positions, and even to change how something is viewed.

There are two complementary points of view for describing object movement. The first is that the object itself is moved relative to a stationary coordinate system or background. The mathematical statement of this viewpoint is described by geometric transformations applied to each point of the object. The second point of view holds that the object is held stationary while the coordinate system is moved relative to the object. This effect is attained through the application of coordinate transformations. An example involves the motion of an automobile against a scenic background. We can also keep the automobile fixed while moving the backdrop fixed (a geometric transformation). We can also keep the automobile fixed while moving the backdrop scenery (a coordinate transformation). In some situations, both methods are employed.

Coordinate transformations play an important role in the instancing of an object – the placement of objects, each of which is defined in its own coordinate system, into an overall picture or design defined with respect to a master coordinate system.

4.2 Geometric Transformations

An object in the plane is represented as a set of points (vertices). Let us impose a coordinate system on a plane. An object Obj in the plane can be considered as a set of points. Every object point P has coordinates (x, y), and so the object is the sum total of all its coordinate points. If the object is moved to a new position, it can be regarded as a new object , all of whose coordinate point P’ can be obtained from

Points in 2-dimensional space will be represented as column vectors:
We are interested in three types of transformation:

- Translation
- Scaling
- Rotation
- Mirror Reflection

4.2.1 Translation

In translation, an object is displaced a given and direction from its original position. If the displacement is given by the vector ,the new object point can be found by applying the transformation T)'y,'x('P$_v$ to P(x, y).

Rotation about the origin

In rotation, the object is rotated °θabout the origin. The convention is that the direction of rotation is counterclockwise if θ is a positive angle and clockwise if θ is a negative angle.

Scaling is the process of expanding or compressing the dimension of an object. Positive scaling constants and , are used to describe changes in length with respect to the x direction and *y* direction, respectively. A scaling constant greater than one indicates an expansion of length, and less than one, compression of length. The scaling transformation is given by xSySysxsS)('*PSPyxss*=where x.s'xx=and . Notice that after a scaling transformation is performed, the new object is located at a different position relative to the origin. In fact, in a scaling transformation the only point that remains fixed is the origin.

Coordinate Transformations
Suppose that we have two coordinate systems in the plane. The first system is located at origin O and has coordinate axes xy figure 4.6. The second coordinate system is located at origin and has coordinate axes Now each point in the plane has two coordinate descriptions: (x, y) or , depending on which coordinate system is used. If we think of the second system as arising from a transformation applied to the first system xy, we say that a coordinate transformation has been applied. We can describe this transformation by determining how the coordinates of a point P are related to the (x, y) coordinates of the same point.

Shear Transformation
The shear transformation distorts an object by scaling one coordinate using the other. If distorts the shape of an object in such a way as if the object were composed of internal layers that has been caused to slide over each other is called shear. Two common shearing transformations are those that shift coordinate x values and those that shift y values.

In very basic two dimensional graphics usually use device coordinates. If any graphics primitive lies partially or completely outside the window then the portion outside will not be drawn. It is clipped out of the image. In many situations we have to draw objects whose dimensions are given in units completely incompatible with the screen coordinates system. Programming in device coordinates is not very convenient since the programmer has to do any required scaling from the coordinates natural to the application to device coordinates. This has led to two dimensional packages being developed which allow the application programmer to work directly in the coordiate system which is natural to the application. These user coordinates are usually called **World Coordinates (WC).** The packages then coverts the coordinates to **Device Coordinates (DC)** automatically. The transformation form the WC to DC is often carried out in tow steps. First using the **Normalisation Transformation** and then the **Workstation Transformation**. The **Viewing Transformation** is the process of going form a window in World coordinates to viewport in **Physical Device Coordinates (PDC).**

Two – Dimensional Viewing and Clipping

Much like what we see in real life through a small window on the wall or the viewfinder of a camera, a Computer-generated image often depicts a partial view of a large scene. Objects are placed into the scene by modeling transformations to a master coordinate system, commonly referred to as the world coordinate system (WCS). A rectangular window with its edge parallel to the axes of the WCS is used to select the portion of the scene for which an image is to be generated (see Fig. 5.2). Sometimes an additional coordinate system called the viewing coordinate system is introduced to simulate the effect of moving and / or tilting the camera.

On the other hand, an image representing a view often becomes part of a larger image, like a photo on an album page, which models a computer monitor's display area. Since album pages vary and monitor sizes differ from one system to another, we want to introduce a device-independent tool to describe the display area. This tool is called the normalized device coordinate system (NDCS) in which a unit (1 x 1) square whose lower left corner is at the origin of the coordinate system defines the display area of a virtual display device. A rectangular viewport with its edges parallel to the axes of the NDCS is used to specify a sub-region of the display area that embodies the image.

The process that converts object coordinates in WCS to normalized device coordinate is called window–to– viewport mapping or normalization transformation. The process that maps normalized device coordinates to Physical Device Co-ordinates (PDC) / image coordinates is called work, station transformation, which is essentially a second window-to-viewport mapping. with a workstation window in the normalized device coordinate system and a workstation viewport in the device coordinate window in the normalized device coordinate system and a workstation viewport in the device coordinate system. Collectively, these two coordinate mapping operations are referred to as viewing transformation

Line Clipping

Lines that do not intersect the clipping window are either completely inside the window or completely outside the window. On the other hand, a line that intersects the clipping window is divided by the intersection point(s) into segments that are either inside or outside the window. The following algorithms provide efficient ways to decide the spatial relationship between an arbitrary line and the clipping window and to find intersection point(s).

Cohen Sutherland Algorithm

The algorithm employs an efficient procedure for finding the category of a line. It proceeds in two steps:

1. Assign a 4-bit region code to each endpoint of the line. The code is determined according to which of the following nine regions of the plane the endpoint lies in Starting from the leftmost bit, each bit of the code is set to true (1) or false (0) according to the scheme

Bit 1 ≡ endpoint is above the window = sign $(y – y_{max})$

Bit 2 ≡ endpoint is below the window = sign $(y_{min}-y)$

Bit 3 ≡ endpoint is to the right of the window = sign $(x - x_{max})$

Bit 4 ≡ endpoint is to the left of the window = sign $(x_{min} - x)$

We use the convention that sign (a)=1 if a is positive, 0 otherwise. Of course, a point with code 0000 is inside the window.

2. The line is visible if both region codes are 0000, and not visible if the bitwise logical AND of the codes is not 0000, and a candidate for clipping if the bitwise logical AND of the region codes is 0000.

For a line in category 3 we proceed to find the intersection point of the line with one of the boundaries of the clipping window, or to be exact, with the infinite extension of one of the boundaries. We choose an endpoint of the line, say (x_1, y_1), that is outside the window, i.e., whose region code is not 0000. We then select an extended boundary line by observing that those boundary lines that are candidates for intersection are of ones for which the chosen endpoint must be "pushed across" so as to change a "1" in its code to a "0" (see Fig. 5.4). This means:

If bit 1 is 1, intersect with line $y = y_{max}$.

If bit 2 is 1, intersect with line $y = y_{min}$.

If bit 3 is 1, intersect with line $x = x_{max}$.

If bit 4 is 1, intersect with line $y = y_{min}$.

If endpoint C is chosen, then the bottom boundary line $y=y_{min}$ is selected for computing intersection. On the other hand, if endpoint D is chosen, then either the top boundary line $y=y_{max}$ or the right boundary line $x = x_{max}$ is used. The coordinates of the intersection point are if the boundary line is vertical.

Area Clipping

An algorithm that clips a polygon must deal with many different cases. The case is particularly note worthy in that the concave polygon is clipped into two separate polygons. All in all, the task of clipping seems rather complex. Each edge of the polygon must be tested against each edge of the clip rectangle; new edges must be added, and existing edges must be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon. We need an organized way to deal with all these cases.

Sutherland-Hodgman's polygon-clipping algorithm

- Polygons can be clipped against each edge of the window one at a time. Windows/edge intersections, if any, are easy to find since the X or Y coordinates are already known.
- Vertices which are kept after clipping against one window edge are saved for clipping against the remaining edges.
- Note that the number of vertices usually changes and will often increases.
- We are using the Divide and Conquer approach.

Four Cases of polygon clipping against one edge

The clip boundary determines a visible and invisible region. The edges from vertex i to vertex i+1 can be one of four types:

- Case 1 : Wholly inside visible region - save endpoint
- Case 2 : Exit visible region - save the intersection
- Case 3 : Wholly outside visible region - save nothing
- Case 4 : Enter visible region - save intersection and endpoint