## 1. Explain following input devices from the context of computer graphics

Mouse

A mouse is a small device that a computer user pushes across a desk surface in order to point to a place on a display screen and to select one or more actions to take from that position.

A mouse consists of a metal or plastic housing or casing, a ball that sticks out of the bottom of the casing and is rolled on a flat surface, one or more buttons on the top of the casing, and a cable that connects the mouse to the computer. As the ball is moved over the surface in any direction, a sensor sends impulses to the computer that causes a mouse-responsive program to reposition a visible indicator (called a cursor) on the display screen. The positioning is relative to some variable starting place. Viewing the cursor's present position, the user readjusts the position by moving the mouse.

The most conventional kind of mouse has two buttons on top: the left one is used most frequently.

Different types of mouse-

Mechanical mouse

Optical and laser mouse

Trackball mouse

Ergonomic mouse

Cordless 3-D mouse

Joystick

A joystick is a personal computer peripheral or general control device consisting of a handheld stick that pivots about one end and transmits its angle in two or three dimensions to a computer.

Most joysticks are two-dimensional, having two axes of movement (similar to a mouse), but three dimensional joysticks do exist. A joystick is generally configured so that moving the stick left or

right signals movement along the X axis, and moving it forward (up) or back (down) signals movement along the Y axis. In joysticks that are configured for three-dimensional movement,

Twisting the stick left (counter-clockwise) or right (clockwise) signals movement along the Z axis. These three axes - X, Y and Z .Joysticks are often used to control games, and usually have one or more push-buttons whose state can also be read by the computer. Most I/O interface cards for PCs have a joystick (game control) port.

The joystick has been the principal flight control in the cockpit of many aircraft, particularly military fast jets, where center stick or side-stick location may be employed.

Joysticks are also used for controlling machines such as cranes, trucks, underwater unmanned vehicles and zero turning radius lawn mowers. Miniature finger-operated joysticks have been adopted as input devices for smaller electronic equipment such as mobile phone.

Graphic Tablet

Graphic tablets are input devices that provide additional flexibility to computer users. Unlike the mouse or keyboard, the graphic tablet serves a wide range of functions. From artwork to writing, graphic tablets are versatile tools that allow for new types of data entry. A graphics tablet works using an X-Y matrix: a grid of vertical and horizontal coordinates. Typically tablet can be used in one of two main modes: "digitizer mode," where the movements of your cursor correspond to points on the screen; and "mouse mode," where the computer generates the position of the pointer on the screen relative to the starting position of the cursor on the pad.

Graphics tablet are used for computer graphics, game design, website graphics design, calligraphy on the computer, CAD and technical drawings and digital photography work.

Different types of Tablet-

Passive tablets

Active tablets

Optical tablets

Acoustic tablets

Electromagnetic tablets

Capacitive tablets

Light Pen

A light-sensitive input device shaped like a pen, used to draw on the computer screen. As the tip of the light pen makes contact with the screen, it sends a signal back to the computer containing the x-y coordinates of the pixels at that point. Light pens can be used on any size screen. Light pens give the user the full range of mouse capabilities, without the use of a pad or any horizontal surface. Users can interact more with the application, in such modes as drag and drop, or highlighting. Light pens target specific pixels, while touch screens activate finger-sized "buttons". Pens offer more active and accurate interaction between the user and the system.

Advantages of the light pen

-Allows you to select objects on a display screen

- It has great accuracy

-drawing directly on the screen so it is more accurate

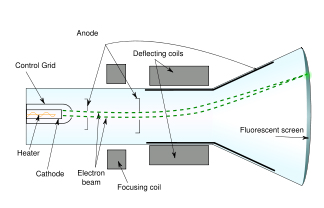
-more precise

Track Ball

A trackball is a [pointing device](http://en.wikipedia.org/wiki/Pointing_device) consisting of a [ball](http://en.wikipedia.org/wiki/Ball) held by a socket containing sensors to detect a rotation of the ball about two axes—like an upside-down [mouse](http://en.wikipedia.org/wiki/Computer_mouse) . The user can move the ball with their thumb or fingers, and sensors inside the case detect which direction the ball is moving. Unlike a mouse, however, the case doesn't move. The user simply rests a hand on it and uses the thumb to move the ball around. Buttons on the side allow users to click on things on screen.  
Large trackballs are common on [CAD](http://en.wikipedia.org/wiki/Computer-aided_design) workstations for easy precision. The trackball mouse is more beneficial because it allows users to maneuver the cursor without requiring movement of the entire mouse. This reduces strain on the user's wrist, hands, arms, and shoulders. Trackball users can also operate the buttons more easily without accidentally moving the mouse itself. Trackball mice also use less space than a traditional mouse, and do not require a mouse pad.

## 2. Explain the working principle of CRT with raster scan graphics?

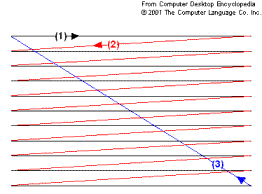
A CRT consists of three basic parts: the electron gun assembly, the phosphor viewing surface, and the glass envelope. The electron gun assembly consists of a heated metal cathode surrounded by a metal anode. The cathode is given a negative electrical voltage and the anode a positive voltage. Electrons from the cathode flow through a small hole in the anode to produce a beam of electrons. The electron gun also contains electrical coils or plates which accelerate, focus, and deflect the electron beam to strike the phosphor viewing surface in a rapid side-to-side scanning motion starting at the top of the surface and working down. The phosphor viewing surface is a thin layer of material which emits visible light when struck by the electron beam. The chemical composition of the phosphor can be altered to produce the colors white, blue, yellow, green, or red. The glass envelope consists of a relatively flat face plate, a funnel section, and a neck section. The phosphor viewing surface is deposited on the inside of the glass face plate, and the electron gun assembly is sealed into the glass neck at the opposite end. The purpose of the funnel is to space the electron gun at the proper distance from the face plate and to hold the glass envelope together so that a vacuum can be achieved inside the finished tube.



500,000 perforations and 1.5 million The CRT used in a color television or color computer monitor has a few additional parts. Instead of one electron gun there are three—one for the red color signal, one for blue, and one for green. There are also three different phosphor materials used on the viewing surface—again, one for each color. These phosphors are deposited in the form of very small dots in a repeated pattern across the screen—red, blue, green, red, blue, green, and so on. The key to a color CRT is a piece of perforated metal, known as the shadow mask, which is placed between the electron guns and the viewing screen. The perforations in the shadow mask are aligned so that the red gun can fire electrons at only the phosphor dots which produce the red color, the blue gun at the blue dots, and the green gun at the green dots. By controlling the intensity of the beam for each color as it scans across the screen, different colors can be produced on different areas of the screen, thus producing a color image. To give an idea of how small the perforations and dots have to be, a 25-inch (63 cm) color television picture tube may have a shadow mask with individual phosphor dots.

Scanning pattern

In raster scanning, the beam sweeps horizontally left-to-right at a steady rate, then blanks and rapidly moves back to the left, where it turns back on and sweeps out the next line. During this time, the vertical position is also steadily increasing (downward), but much more slowly — there is one vertical sweep per image frame, but one horizontal sweep per line of resolution. Thus each scan line is sloped slightly "downhill" (towards the lower right), with a slope of approximately –1/horizontal resolution, while the sweep back to the left (retrace) is significantly faster than the forward scan, and essentially horizontal. The resulting tilt in the scan lines is very small, and is dwarfed in effect by screen convexity and other modest geometrical imperfections.



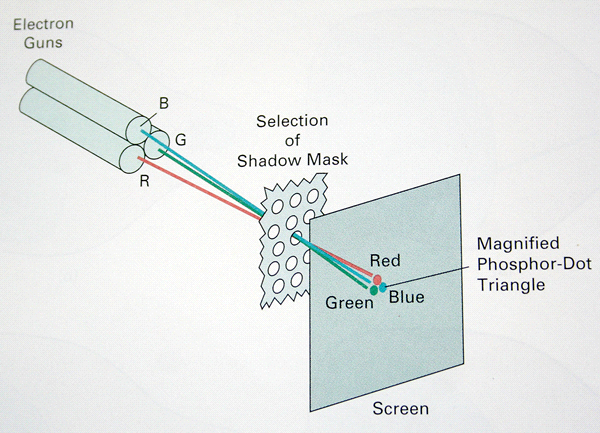
Furthermore, wide-deflection-angle CRTs need horizontal sweeps with current that changes proportionally faster toward the center, because the center of the screen is closer to the deflection yoke than the edges. A linear change in current would swing the beams at a constant rate angularly; this would cause horizontal compression toward the center. Displaying or capturing a video image line by line. Computer monitors and TVs use this method whereby electrons are beamed (scanned) onto the phosphor coating on the screen a line at a time from moved back to the left and down one line, which is known as the "horizontal retrace." When the bottom-right corner is reached, the gun is returned to the top-left corner, known as the "vertical retrace." For TV signals, these "fly back" periods in which the electron beam is moved to a different line are also called the "horizontal" and "vertical blanking intervals."

## 3. Explain the working principle of shadow mask CRT with the help of a diagram?

The shadow mask is one of two major technologies used to manufacture cathode ray tube (CRT) televisions and computer displays that produce color images. The other approach is aperture grille, better known by its trade name, Trinitron. All early color televisions and the majority of CRT computer monitors used shadow mask technology. Both of these technologies are largely obsolete, having been increasingly replaced since the 1990s by the Liquid Crystal Display (LCD).

A shadow mask is a metal plate punched with tiny holes that separate the colored phosphors in the layer behind the front glass of the screen. Three electron guns at the back of the screen sweep across the mask, with the beams only reaching the screen when they pass over the holes. As the guns are physically separated at the back of the tube, their beams approach the mask from three slightly different angles, so after passing through the holes they hit slightly different locations on the screen. The screen is patterned with dots of colored phosphor positioned so they can only be hit by the beam from only one of the guns passing through only one of the holes. For instance, a particular spot on the screen can only be hit by the beam from the "blue gun" passing through a particular hole in the mask. This arrangement allows the colored guns to address individual dots on the screen, even though their beams are much too large and too poorly aimed to do so without the mask in place.

The red, green, and blue phosphors for each pixel are generally arranged in a triangular shape (sometimes called a "triad"). For television use, modern displays (starting in the late 1960s) use rectangular slots instead of circular holes, improving brightness.



The guns, arranged in a triangle at the back of the tube, were aimed to focus on the metal plate and scanned it as normal. For much of the time during the scan, the beams would hit the back of the plate. However, when the beams passed a hole they would briefly pass through it to the phosphor in front of the plate. In this way, the plate ensured that the beams were perfectly aligned with the colored phosphor dots. This still left the problem of focusing on the correct colored dot. Normally the beams from the three guns would each be large enough to light up all three colored dots on the screen. The mask helped by mechanically attenuating the beam to a small size just before it hit the screen.

But the real genius of the idea is that the beams approached the metal plate from different angles. After being cut off by the mask, the beams would continue forward at slightly different angles, hitting the screens at slightly different locations. The spread was a function of the distance between the guns at the back of the tube, and the distance between the mask plate and the screen. By painting the colored dots at the correct locations on the screen, and leaving some room between them to avoid crosstalk, the guns would be guaranteed to hit the right colored spot.

Although the system was simple, it had a number of serious practical problems.

As the beam swept the mask, the vast majority of its energy was deposited on the mask, not the screen in front of it. A typical mask of the era might have only 15% of its surface open. To produce an image as bright as the one on a traditional B&W television, the electron guns in this hypothetical shadow mask system would have to be five times more powerful. Additionally, the dots on the screen were deliberately separated in order to avoid being hit by the wrong gun, so much of the screen was black.[13] This required even more power in order to light up the resulting image. And as the power was divided up among three of these much more powerful guns, the cost of implementation was much higher than a similar B&W set.

The amount of power deposited on the screen was so great that thermal loading was a serious problem. The energy the shadow mask absorbs from the electron gun in normal operation causes it to heat up and expand, which leads to blurred or discolored images . Signals that alternated between light and dark caused cycling that further increased the difficulty of keeping the mask from warping.

Furthermore, the geometry of the system required complex systems to keep the images in focus. If you consider the beam when it is sweeping across the middle area of the screen, the three beams from the individual guns are each traveling the same distance and meet the holes in the mask at equal angles. In the corners of the screen some beams have to travel further and all of them meet the hole at a different angle than at the middle of the screen. These issues required additional electronics and adjustments to keep the signal in focus.

## 4. Explain how does the computer graphic system works with the help of a block diagram?



A computer graphics system is a computer system which has all the components of a general purpose computer system. 5 major elements in the system are:

1. I/p devices

2. Processor

3. Memory

4. Frame buffer

5. O/p devices

This model is general enough to include workstations and personal computer, interactive game systems, and sophisticated image-generation systems.

Pixels are stored in a part of memory called the frame buffer**.** The frame buffer can be viewed as the core element of a graphics system. Its resolution determines the details that can be seen in the image. The depth**,** or precision, of the frame buffer, defined as the number of bits that are used for each pixel determines properties such as how many colors can be represented on a given system.

The frame buffer usually is implemented with special types of memory chips that enable fast redisplay of the contents of the frame buffer. In s/w based systems, such as those used for high-resolution rendering or for generating complex visual effects that cannot produced in real time, the frame buffer is part of system memory.

In a simple system, there may be only one processor, the CPU of the system, which must do both the normal processing and the graphical processing. The main graphical function of the processor is to take specifications of graphical primitives (such as lines, circles, and polygons) generated by application programs and to assign values to the pixels in the frame buffer that best represent these entities. For example, a triangle is specified by its 3 vertices, but to display its outline by the three line segments connecting the vertices, graphics system must generate a set of pixels that appear as line segments to the viewer. The conversion of geometric entities to pixel colors and locations in the frame buffer is known as rasterization, or scan conversion. In early graphics systems, the frame buffer was part of the standard memory that could be directly addressed by the CPU. Today, virtually all graphics systems are characterized by special-purpose GPU, custom-tailored to carry out specific graphics functions. The GPU can be either on the motherboard of the system or on a graphics card. The frame buffer is accessed through the GPU and may be included in the GPU.

**O/P devices**

For many years, the dominant type of display (or monitor) has been the cathode ray tube (CRT). Although various flat-panel technologies are now more popular, the basic functioning of the CRT has much in common with these newer displays. When electrons strike the phosphor coating on the tube, light is emitted. The direction of the beam is controlled by two pairs of deflection plates. The output of the computer is converted, by digital-to-analog converters, to voltages across the x and y deflection plates. Light appears on the surface of the CRT when a sufficiently intense beam electrons is directed at the phosphor.

If the voltages steering the beam change at constant rate, the beam will trace a straight line, visible to a viewer. Such a device is known as the random-scan, calligraphic, or vectorCRT, because the beam can be moved directly from any position to any another position. If the intensity of the beam is turned off, the beam can be moved to a new position without changing any visible display.

A typical CRT will emit light for only a short time—usually a few milliseconds—after the phosphor is excited by the electron beam. For a human to see a steady, flicker image

on most CRT displays, the same path must be retraced, or refreshed, by the beam at a sufficiently high rate, the refresh rate**.** In older systems, the refresh rate is determined by the frequency of the power system, 60 cycles/second or 60Hz in the United States and 50Hz in the much of the rest of the world.

In a raster system, the graphics system takes pixels from the frame buffer and displays them as point on the surface of the display in one of two fundamental ways. In a no interlacedor progressivedisplay, the pixels are displayed row by row, or scan line by scan line, at the refresh rate. In an interlaceddisplay, odd rows and even rows are refreshed alternatively. Interlaced displays are used in commercial television. In an interlaced display operating at 60Hz, the screen is redrawn in its entirely only 30 times/second, although the visual system is tricked into thinking the refresh rate is 60Hz rather than 30Hz.

**I/P devices**

Most graphics systems provide a keyboard and at least one I/p device. The most common I/p devices are the mouse, the joystick, and the data tablet. Each provides positional information to the system, and each usually is equipped with one or more buttons to provide signals to the processor. Often calledpointing devices, these devices allow a user to indicate a particular location on the display.

## 5. Define

### Pixel:

In digital imaginary, a pixel is a single point in a raster image. The pixel is the smallest addressable screen element. It is the smallest unit of picture that can be controlled. The address of a pixel is its own co-ordinates. Each pixel is a sample of an original image; more samples provided more accurate representation of the original. The number of pixels in an image is called resolution. Pixels are used as a measure of resolution (dpi,ppi).

### Voxel:

A voxel is a volume element representing a value .On a rectangular grid in 3D space. As with pixels in a bit-map, voxels do not have their position encoded with their. The position of a voxel is based on a position of relative to other voxel. It is good at representing regularly-sampled, homogeneously filled visualization and analysis.

### CGA:

Color graphic adapter introduced in 1981.It was IBM’s First color graphics card and the first color computer display standard for PC.The standard IBM CGA was equipped with 16Kb.Of video memory and cold be connected either to a NTSC compatible monitor or TV via an RCA jack.

### VGA:

Video graphics array refers specifically to the display hardware introduced with IBM in 1987.It has a resolution of 640\*480.VGA was the last graphical standard introduced by IBM.VGA was super sided by numerous extensions of VGA collectively called as Super VGA.

### SVGA:

It is a broad term that covers wide range of computer display system standards.Svga refers to a resolution of 800\*600 pixels. Its first version called for 4-bit pixels. It was expand to 1024\*768 8bit pixels.

### Nvidia:

Nvidia is a multinational corporation which specializes in the development of graphics processing units and chipset technologies for workstations, personal computers, and mobile devices. Based in Santa Clara, California, the company has become a major supplier of integrated circuits (ICs), designing graphics processing units (GPUs) and chipsets used in graphics cards, in personal-computer motherboards, and in video game consoles. Nvidia graphic cards are compatible with Open GL, Direct X and AGP technology.

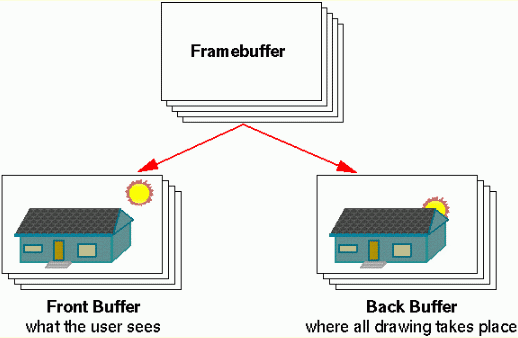
## 6. Write a note on the concept of double frame buffer for visualization of dynamic data.

In computer graphics, double buffering is a technique for drawing graphics that shows no (or less) flicker, tearing, and other artifacts.

It is difficult for a program to draw a display so that pixels do not change more than once. For instance to update a page of text it is much easier to clear the entire page and then draw the letters than to somehow erase all the pixels that are not in both the old and new letters. However, this intermediate image is seen by the user as flickering. In addition computer monitors constantly redraw the visible video page (at around 60 times a second), so even a perfect update may be visible momentarily as a horizontal divider between the "new" image and the un-redrawn "old" image, known as tearing.

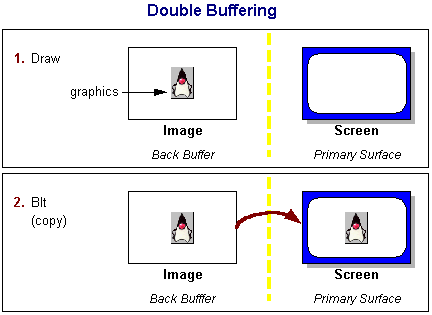
Double buffering is a popular approach to synchronizing the rendering and hardware refresh rates. Most modern APIs support double buffering. Basically such systems have two frame buffers: the front buffer and the back buffer.

The front and back buffers represent a double-buffered frame buffer. The front buffer is, more or less, what you see on the screen. The back buffer is the image that is typically rendered to.





The front buffer is displayed while the application renders into the back buffer. When the rendering to the back buffer is completed, the application requests a buffer swap from the graphics display hardware, this effectively transfers the back buffer data into the front buffer, the result is that the former back buffer is the one that is displayed (it becomes the new front buffer), and the previous front buffer becomes the new back buffer.



Another figure that shows the concept of double buffering, Blt refers to block transfer.

A software implementation of double buffering has all drawing operations store their results in some region of system RAM; any such region is often called a "back buffer". When all drawing operations are considered complete, the whole region (or only the changed portion) is copied into the video RAM (the "front buffer"); this copying is usually synchronized with the monitor's raster beam in order to avoid tearing. Double buffering necessarily requires more video memory and CPU time than single buffering because of the video memory allocated for the back buffer, the time for the copy operation, and the time waiting for synchronization.

## 7. Draw the polygon, rectangular graphics primitives with closed boundary

### Graphics Primitives

A primitive is a graphics object that is essential for the creation or construction of complex images.

### Polygon

A polygon, even though generally constructed from straight lines, is an important graphics primitive. So often we want to handle polygon as a single entity, as images of objects from the real world consist in large, part of polygons.

A polygon is a closed area of image bounded by straight or curved lines and filled with one solid colour. Since images are two dimensional, a polygon is a closed planar figure.

Implementing a polygon as a graphics primitive is natural and helpful. We can define polygon as an image which consists of a finite ordered set of straight boundaries called edges. Alternately, the polygon can be defined by an ordered sequence of vertices, the corners of the polygon.

Polygon[{pt1,pt2,…..}]

This is a graphics primitive that represents a filled polygon.

Polygon[{{pt11, pt22,…..},{pt21,…..}]

Represents a collection of polygons,

Basic Examples

Triangles:

|  |  |
| --- | --- |
| [In[1]:=](javascript:input('i_1')) | [Description: Click for copyable input](javascript:input('i_1) |

|  |  |  |
| --- | --- | --- |
| Out[1]= | | Description: http://reference.wolfram.com/mathematica/ref/Files/Polygon.en/O_1.gif |
| [In[2]:=](javascript:input('i_3')) | [Description: Click for copyable input](javascript:input('i_3) | | |

|  |  |
| --- | --- |
| Out[2]= | Description: http://reference.wolfram.com/mathematica/ref/Files/Polygon.en/O_2.gif |

Self-intersecting polygon:

|  |  |
| --- | --- |
| [In[1]:=](javascript:input('i_5')) | [Description: Click for copyable input](javascript:input('i_5) |

|  |  |
| --- | --- |
| Out[1]= | Description: http://reference.wolfram.com/mathematica/ref/Files/Polygon.en/O_3.gif |

Differently styled 2D polygons:

|  |  |
| --- | --- |
| [In[1]:=](javascript:input('i_7')) | [Description: Click for copyable input](javascript:input('i_7) |

|  |  |
| --- | --- |
| [In[2]:=](javascript:input('i_9')) | [Description: Click for copyable input](javascript:input('i_9) |

|  |  |
| --- | --- |
| Out[2]= | Description: http://reference.wolfram.com/mathematica/ref/Files/Polygon.en/O_4.gif |

Differently styled 3D polygons:

|  |  |
| --- | --- |
| [In[1]:=](javascript:input('i_11')) | [Description: Click for copyable input](javascript:input('i_11) |

|  |  |
| --- | --- |
| [In[2]:=](javascript:input('i_13')) | [Description: Click for copyable input](javascript:input('i_13) |

|  |  |
| --- | --- |
| Out[2]= | Description: http://reference.wolfram.com/mathematica/ref/Files/Polygon.en/O_5.gif |

Rectangle

Rectangle [{xmin,ymin},{xmax,ymax}]  
is a two-dimensional graphics primitive that represents a filled rectangle, oriented parallel to the axes.

Rectangle [{xmin,ymin}]  
corresponds to a unit square with its bottom-left corner at [{xmin,ymin}].

Basic Examples

A unit square:

|  |  |
| --- | --- |
| [In[1]:=](javascript:input('i_1')) | [Description: Click for copyable input](javascript:input('i_1) |

|  |  |
| --- | --- |
| Out[1]= | Description: http://reference.wolfram.com/mathematica/ref/Files/Rectangle.en/O_1.gif |

 Two squares:

|  |  |
| --- | --- |
| [In[1]:=](javascript:input('i_3')) | [Description: Click for copyable input](javascript:input('i_3) |

|  |  |
| --- | --- |
| Out[1]= | Description: http://reference.wolfram.com/mathematica/ref/Files/Rectangle.en/O_2.gif |

 Various rectangles:

|  |  |
| --- | --- |
| [In[1]:=](javascript:input('i_5')) | [Description: Click for copyable input](javascript:input('i_5) |

|  |  |
| --- | --- |
| Out[1]= | Description: http://reference.wolfram.com/mathematica/ref/Files/Rectangle.en/O_3.gif |

 Differently styled rectangles:

|  |  |
| --- | --- |
| [In[1]:=](javascript:input('i_7')) | [Description: Click for copyable input](javascript:input('i_7) |

|  |  |
| --- | --- |
| Out[1]= | Description: http://reference.wolfram.com/mathematica/ref/Files/Rectangle.en/O_4.gif |

## 8. Distinguish between random and raster scan display devices with help of neat diagram.

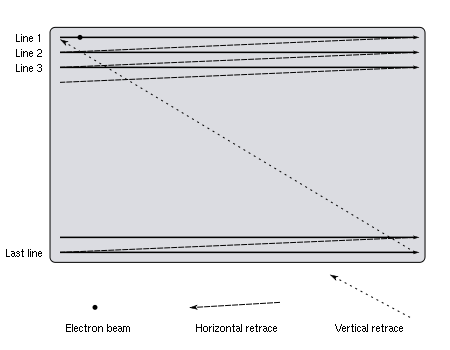
### Raster Scan

A raster scan, or raster scanning, is the rectangular pattern of image capture and reconstruction in television.

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 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).

Refreshing on raster-scan displays is carried out at the rate of 60 to 80 frames per second, although some systems are designed for higher refresh rates. Sometimes, refresh rates are described in units of cycles per second, or Hertz (Hz), where a cycle corresponds to one frame. At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line. The return to the left of the screen, after refreshing each scan line, is called the horizontal retrace of the electron beam. And at the end of each frame (displayed in 1/80th to 1/60th of a second), the electron beam returns (vertical retrace) to the top left corner of the screen to begin the next frame.

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 retrace, the beam sweeps out the remaining scan lines. 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.

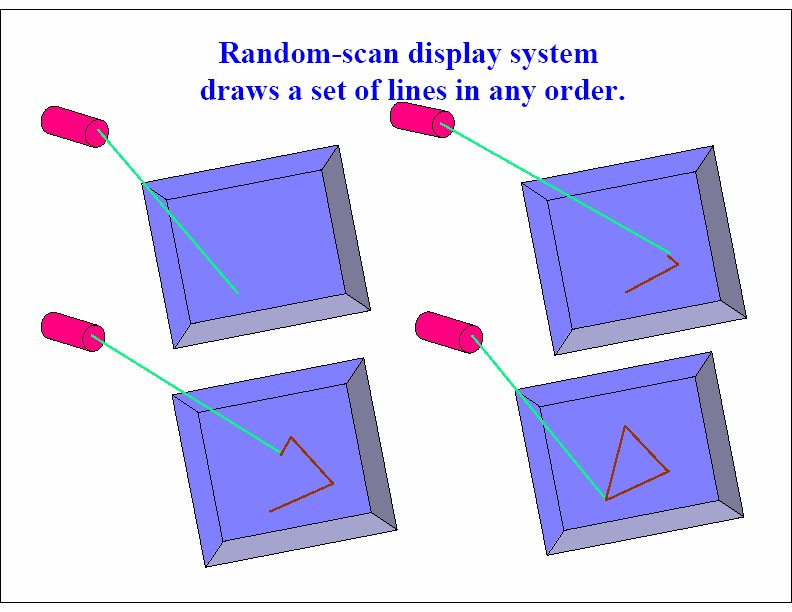


### Random Scan

In Random Scan System, an electron beam is directed to only those 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 vectordisplays (or stroke-writingor calligraphicdisplays). The component lines of a picture can be drawn and refreshed by a random-scan system in any specified order.

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. Sometimes the refresh display fileis called the display list**,** display program, or simply the refresh buffer. To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn.

After all line- drawing commands have been processed, the system cycles back to the first line command in the list. Random-scan displays are designed to draw al the component lines of a picture 30 to 60 times each second.



## Differences between Raster Scan and Random Scan

|  |  |  |
| --- | --- | --- |
| **Base of Difference** | **Raster Scan System** | **Random Scan System** |
| **Electron Beam** | The electron beam is swept across the screen, one row at a time, from top to bottom. | The electron beam is directed only to the parts of screen where a picture is to be drawn. |
| **Resolution** | Its resolution is poor because raster system in contrast produces zig-zag lines that are plotted as discrete point sets. | Its resolution is good because this system produces smooth lines drawings because CRT beam directly follows the line path. |
| **Picture**  **Definition** | Picture definition is stored as a set of intensity values for all screen points, called pixels in a refresh buffer area. | Picture definition is stored as a set of line drawing instructions in a display file. |
| **Realistic Display** | The capability of this system to store intensity values for pixel makes it well suited for the realistic display of scenes contain shadow and color pattern. | These systems are designed for line-drawing and can’t display realistic shaded scenes. |
| **Refresh Rate** | Refreshing on raster-scan displays is carried out at the rate of 60 to 80 frames per second | Refreshing on random-scan displays is carried out at the rate of 30 to 60 frames per second |
| **Draw an Image** | Screen points/pixels are used to draw an image. | Mathematical functions are used to draw an image. |
| **Cost** | Raster Scan Systems are less expensive. | Random-scan systems are generally costlier. |

## 9. Explain about NAPLP

NAPLPS (North American Presentation Level Protocol Syntax)

NAPLPS (North American Presentation Level Protocol Syntax) is a graphics language for use originally with videotex and teletext services. NAPLPS was developed from the Telidon system developed in Canada, with a small number of additions from AT&T. The basics of NAPLPS were later used as the basis for several other microcomputer based graphics systems.

ONE WAY SYSTEM

Telidon-based teletext was tested in a few North American trials in the early 1980s -- CBC IRIS, TVOntario, MTS-sponsored Project IDA, to name a few.

NAPLPS was also part of the NABTS teletext standard, for the encoding and display of teletext pages.

In the late 1980s and early 1990s, SportsChannel ran a service called Sports Plus Network, which ran sports news and scores while SportsChannel was not otherwise on the air. The screens, which frequently featured team logos or likenesses of players in addition to text, were drawn entirely with NAPLPS graphics and resembled the loading of Prodigy pages over a modem, though slightly faster.

TWO WAY SYSTEMS

 Various two-way systems using NAPLPS appeared in North America in the early 1980s. The biggest North American examples were Knight Ridder's Viewtron (based in Miami) and the Los Angeles Times' Gateway service (based in Orange County). Both used the Sceptre NAPLPS terminal from AT&T. The Sceptre contained a slow modem that connected over the consumer's telephone line to host computers. The Sceptre was expensive whether purchased or rented. Despite huge investments by their parent companies, neither Viewtron nor Gateway lasted into the second half of the decade.

Other early-1980s NAPLPS technology was deployed in Canada, both as a way for rural Canadians to get news and weather information and as the platform for touchscreen information kiosks. In Vancouver these were featured at Expo 86. The kiosks became ubiquitous in Toronto under the name Teleguide, and were deployed in many shopping centres and at major tourist attractions. The latter city was the North American nexus of NAPLPS and the home of Norpak, the most successful of NAPLPS-oriented developers. Norpak created and sold hardware and software for NAPLPS development and display. TVOntario also developed NAPLPS content creation software.

London, Ontario - based Cableshare used NAPLPS as the basis of touch-screen information kiosks for shopping malls, the flagship of which was deployed at Toronto's Eaton Centre. The system relied on an 8085-based microcomputer which drove several NAPLPS terminals fitted with touch screens, all communicating via Datapac to a back end database. The system offered news, weather and sports information along with shopping mall guides and coupons. Cableshare also developed and sold a leading NAPLPS page creation utility called the "Picture Painter."

In the late 1980s, Tribune Media Services (TMS) and the Associated Press operated a cable television channel called AP News Plus that provided NAPLPS-based news screens to cable television subscribers in many U.S. cities. The news pages were created and edited by TMS staffers working on an Atex editing system in Orlando, Florida, and sent by satellite to NAPLPS decoder devices located at the local cable television companies. Among the firms providing technology to TMS and the Associated Press for the AP News Plus channel was Minneapolis-based Electronic Publishers Inc. (1985-1988).

In 1981, two amateur radio operators (VE3FTT and VE3GQW) received special permission from the Canadian Department of Communications to carry out on-air experiments using NAPLPS syntax which was technically not legal at the time because it was a "coded transmission". Following their report on the success of the tests, the DOC then permitted general use of NAPLPS on amateur radioteletype. This was reported in the ARRL Radio Handbook for several years following.

## 10. Explain aliasing and staircase effect

### A) Aliasing

Let us now consider how this model applies to Computer Graphics.

All antialiasing techniques use some form of blurring or smoothening the image to reduce the effect of aliasing.

All rendering algorithms globally fall in two categories.

Analytical

These algorithms can prefilter an image and take out its high frequencies before sampling the pixel values which is an anti-aliasing filtering operation.

Discrete

These algorithms consider the image at regularly spaced sample points. They are simpler and easier to implement.

In the discrete algorithms, the generation and sampling of the image are interwined. There is no continuous image function that is sampled - an image value that is calculated at sample points. We cannot insert an antialiasing filter which operates in the continuous information domain. Another point we need to consider, is that the images which are function of spatial variables over a sampling grid.

Differences that arise in Graphics

The amplitude characteristic of most images peaks at the origin and decreases as the reciprocal of the spatial frequency. This may not be the case with graphics as it may result in high values of F(u,v) which is the transform of the two dimensional point, far away from the origin. We have to distinguish between theoretical transforms and the transforms of images on the screen. The spatial frequency of computer graphics abstractions can extend to infinity in both u & v. Screen images are however limited in each direction by the resolution of the frame store and by the line rate and horizontal resolution of the display device.

The locus of equal spatial frequencies is a circle centered at the origin. So a perfect low pass filter is a cylinder centred at the origin. High frequencies contribute to the sharpness in the image. So removing these blurs the image. In the Fourier domain, the antialiasing operations are a tradeoff between artefact visibility and image blurring because we remove high spatial frequencies.

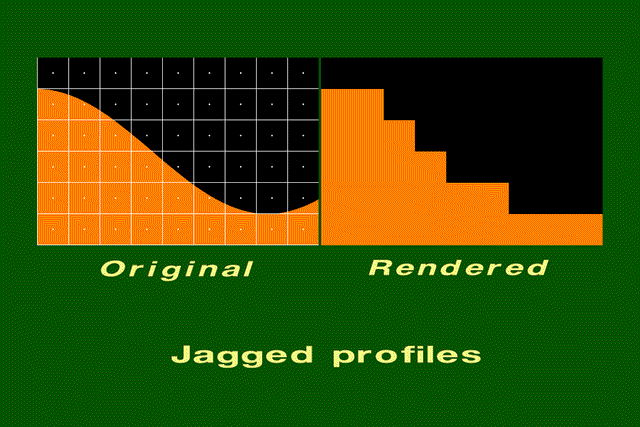
Thus we find that simple filtering cannot be applied in Graphics directly.

Effects caused by aliasing.

The errors caused by aliasing are called artifacts. Common aliasing artefacts include jagged profiles, disappearing or improperly rendered fine detail, and disintegrating textures.

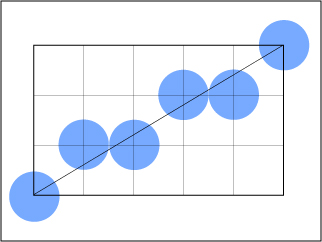
Jagged profiles.

The picture on the left shows the sampling grid superimposed on the original scene. The picture on the right is the rendered image. A jagged profile is quite evident in the rendered image. Also known as "jaggies", jagged silhouettes are probably the most familiar effect caused by aliasing. Jaggies are especially noticeable where there is a high contrast between the interior and the exterior of the silhouette.

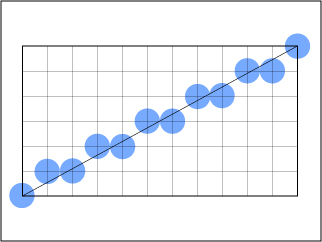


### b) Staircase effect

Consider using the midpoint algorithm to draw a 1-pixel-thick black line, with slope between 0 and 1, on a white background. In each column through which the line passes, the algorithm sets the colour of the pixel that is closest to the line. Each time the line moves between columns in which the pixels closest to the line are not in the same row, there is a sharp jag in the line drawn into the canvas. The same is true for other scan-converted primitives that can assign only one of two intensity values to pixels."



If we now use a display device with a higher horizontal and vertical resolution, the jags are getting smaller but more numerous:



Although the resulting picture looks better, jags still exist and in addition the memory requirement of the picture increases. Increasing resolution is an expensive solution that only diminishes the problem of jaggies, but it does not eliminate the problem itself.

## 11. What are the applications of computer graphics?

The application of computer graphics includes:

Creation of a graphical user interface (gui) – a graphic, mouse – oriented paradigm which allows the user to interface or interact with a computer.

Business presentation graphics

Cartography

Weather maps – real time mapping symbolic representations

Satellite imaging – geodesic images

Medical imaging – MRIs, CAT scans, etc,-inrasive internal examination

Training – flight simulation, computer aided instruction etc.

Simulation & modeling – replacing physical modeling & enactments

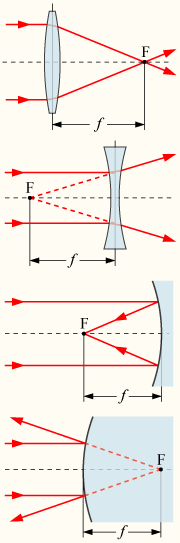
## 12. Explain the principle behind digital camera with help of neat diagram

Cameras work with the light of the visible spectrum. A camera generally consists of some kind of enclosed hollow, with an opening or aperture at one end for light to enter, and a recording or viewing surface for capturing the light at the other end. Most cameras have a lens positioned in front of the camera's opening to gather the incoming light and to focus the image (or part of the image), on the recording surface. The diameter of the aperture is often controlled by a diaphragm mechanism, but some cameras have a fixed-size aperture.

[Digital cameras](http://en.wikipedia.org/wiki/Digital_camera) use electronics, usually a charge coupled device ([CCD](http://en.wikipedia.org/wiki/Charge-coupled_device)) or sometimes a Complementary Metal–Oxide–Semiconductor ([CMOS](http://en.wikipedia.org/wiki/CMOS)) sensor to capture images which can be transferred or stored in computer memory inside the camera for later playback or processing.

An *image sensor* is a device that converts a visual image to an electric signal. It is used in digital cameras (as well as other imaging devices). It is usually an array of charge-coupled devices (CCD) or CMOS sensors such as active pixel sensors.

To reconstruct a color image from the data collected by the color filtering array, you need to fill in the blanks. The mathematics here is subject to individual implementation, and is called [demosaicing](http://en.wikipedia.org/wiki/Demosaicing). If you have a RAW image, you can use different [demosaicing](http://en.wikipedia.org/wiki/Demosaicing) than what is built into the camera, often yielding higher quality.

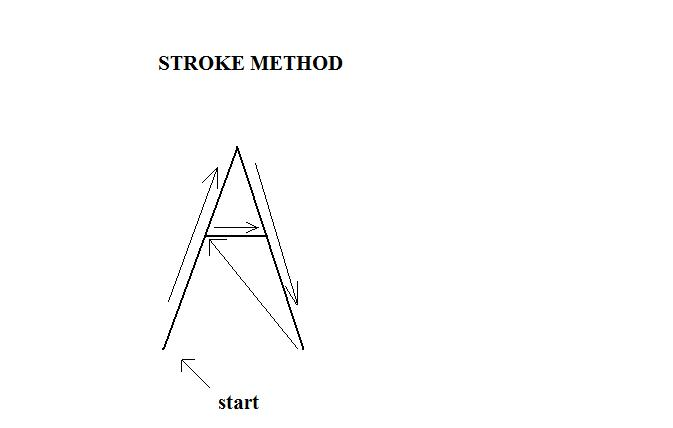
A [camera lens](http://en.wikipedia.org/wiki/Camera_lens) usually has an aperture adjustment mechanism to control the amount of light that may pass. It also has a shutter, to control the length of time during which light may pass through the lens. [](http://upload.wikimedia.org/wikipedia/commons/8/8f/Focal-length.p)

13. Explain character generation methods with help of diagram**.**

**Stroke method to generate character**

This method uses small line segment to generate characters. Small series of line segment are drawn like stroke of a pen to form the characters as shown in the figure .

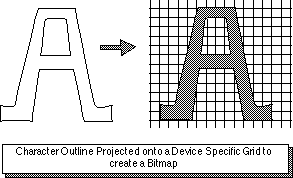
Hence we decide which line segment are these segment using line drawing algorithms we can draw characters and display.



**BITMAP METHOD:-**

It is also called dot matrix format. Because these characters are represent as arrays of dots in the matrix form .

It is a 2d array having row and column (5×7,7×9and 9×13) array are used. Each dot is a pixel. Characters are placed on the screen by copying pixel value from character array into some portion of screen frame buffer. Dot pattern for all characters are store into hardware device called character generator chip.



Character A in dot matrix format

## 14. What are the properties of good line generation algorithm? Explain with help of neat diagram.

***i)*** Line should appear straight: Point plotting techniques are admirably suited to the generation of lines parallel or at 45 degree to the X and Y axes. Other lines cause a problem, a line segment, though it starts and finishes at addressable points, may happen to pass through no other addressable points in between. By applying anti-aliasing techniques, we can minimize the aliasing effects (staircase effects).

***ii)*** Line should terminate accurately: Unless lines are plotted accurately, they may terminate at the wrong place. This effect is seen as a small gap between end point of one line and the starting point of the next or as cumulative errors.

**Fig: Output from a poor line generation algorithm**

***iii)*** Lines should have constant density: With bright lines plotted on a dark background, line density is observed as brightness; when the line is black and the background light; it is seen as blackness. Line density is proportional to the number of dots displayed divided by the length of the line. To maintain constant density, dots should be equally spaced.

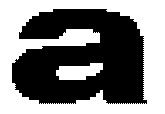
**Fig: Uneven line density by bunching of dots**  **Fig: Even line density**

***iv)*** Line density should be independent of line length and angle: To achieve constant line density we must maintain a constant number of dots per unit length. Before plotting the line, we must determine its exact length, by computing square root. We must control the rate at which dots are plotted. We must compute approximate line-length estimate and use a line-generation algorithm.

***v)*** Line should be drawn rapidly: Line should appear rapidly on the screen. This implies using the minimum of computation to draw the line; this ideally computation should be performed by special purpose-hardware.

## 15. What is anti-aliasing technique and list them? Explain with the help of example.

Anti-Aliasing is a method of fooling the eye that a jagged edge is really smooth. Anti-Aliasing is often referred in games and on graphics cards. In games especially the chance to smooth edges of the images goes a long way to creating a realistic 3D image on the screen. Remember though that Anti-Aliasing does not actually smooth any edges of images it merely fools the eye. Example below to demonstrate the effects of Anti-Aliasing.

The letter on the left is a blown up letter a with no anti-aliasing. The letter on the right has had anti-aliasing applied to it. In this blown up form it looks like its simply blurred but if we reduce the size down to a more standard size you may see the difference.

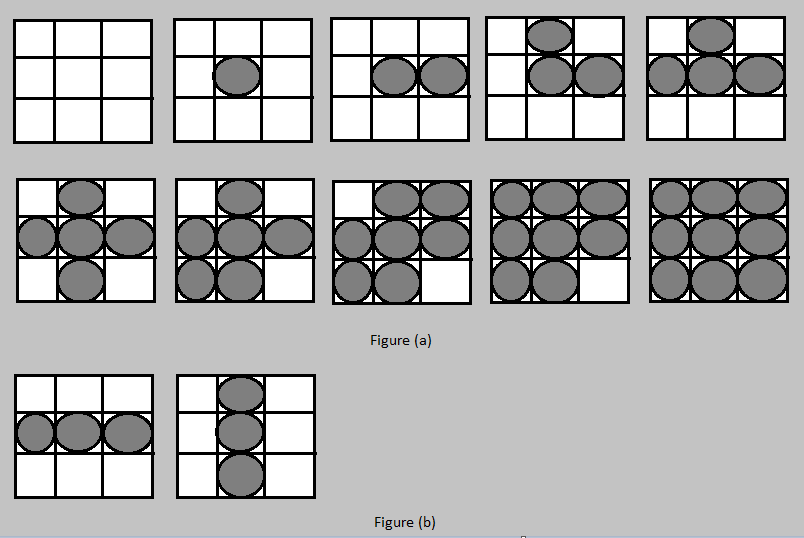
aaliasing2

Now look closely at the two letters. You can still tell that the letter of the left is jagged but the letter on the right looks a lot smoother and less blurry than the example above. Remember I have only shrunk the image down back to normal size and have not altered anything else to the image at all. So as you can see, Anti-Aliasing brings a much more pleasing image to the eye

.

**1. HALFTONING**

Often it is necessary to present a natural image that has pixels of a great many different intensity values on a display with a limited range of output values. Solutions to this problem are called halftoning techniques, developed largely in order of print photographs using black ink on white paper. The basic idea is to use patterns of black and white to give the impression of intermediate intensities. Consider a "pattern technique" that uses a 3 X 3 array of binary pixels to display one of ten intensities. To achieve intensity i(0<= i <=9), we set i of the 9 pixels to white, and 9-i to black. Figure (a) shows a set of ten patterns that might be acceptable for this purpose. Figure (b) shows patterns for i=3 that are less satisfactory as they will produce a pattern of horizontal stripes if repeated many times.



Another class of halftoning techniques is based on the idea of thresholding: if the image intensity of a pixel exceeds a threshold, white is displayed for that pixel, otherwise black. To explain them some additional notation is required.

Suppose the image to be displayed is at *(x,y)*  Has an intensity represented by *I(x,y)*  The intensity ranges between a black value b and a white value w where *b<= I(x,y) <= w.* Let *g = ( b + w )/2* be a threshold in the middle of the range.

MODULATION

The first technique modulates the intensity signal with a signal *M(x, y)* whose values range from -g to g, with average value 0. The pixel at *(x, y)* is made white if *I(x, y) + M(x, y) > g*; otherwise it is set to black. Suitable functions for *M* are sinusoids.

ERROR DISTRIBUTION

It utilizes a algorithm that selects an intensity for display at (x, y) and distributes the error to the three neighbours: three-eighths of the error is propagated to the right neighbor, three-eighths to the bottom neighbour, and one-fourth along the diagonal.

**2. INCREASING THE RESOLUTION**

Consider using the midpoint algorithm to draw a 1-pixel-thick black line, with slope between 0 and 1, on a white background. In each column through which the line passes, the algorithm sets the color of the pixel that is closest to the line. Each time the line moves between columns in which the pixels closest to the line are not in the same row, there is a sharp jag in the line drawn into the canvas, as is clear in figure (a).

Suppose we now use a display device with twice the horizontal and vertical resolution. As shown in figure (b), the line passes through twice as many columns and therefore has twice as many jags, but each jag is half as large as x and y. Although the resulting picture looks better, the improvement comes at the price of quadrupling the memory cost, memory bandwidth, and the scan conversion time.

Increasing the resolution is an expensive solution that only diminishes the problem of jaggies- it does not eliminate the problem.

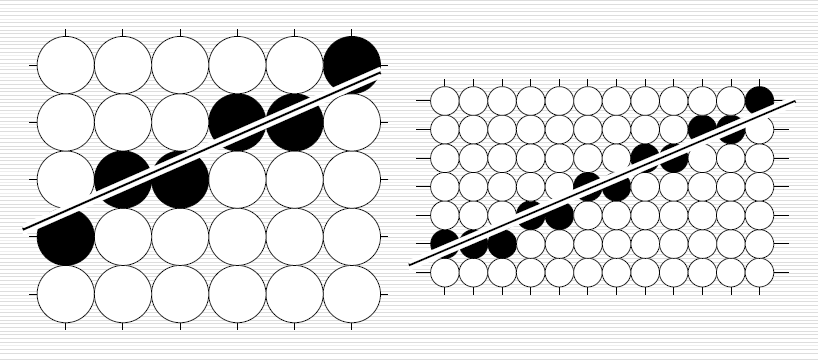


Figure (a) Figure (b)

**3. UNWEIGHTED AREA SAMPLING**

A scan-converted primitive occupies a finite area on the screen-even the thinnest horizontal or vertical line on a display surface is 1 pixel thick and the lines at other angles have width that varies over the primitive. Thus, we think of any line as a rectangle of a desired thickness covering a portion of the grid, as shown in figure (c). It follows that a line should not set the intensity of only a pixel in a column to black, but rather should contribute some amount of intensity to each pixels in the column whose area it intersects. Then, for 1 pixel thick lines, only horizontal and vertical lines would affect exactly 1 pixel in their column or row. For lines at other angles, more than 1 pixel would now be set in a column or row, each to an appropriate intensity. Pixels form an array of no overlapping square tiles covering the screen, centered on grid points. We assume that a line contributes to each pixel’s intensity an amount proportional to the percentage of the pixel's tile it covers. A fully covered pixel on a black and white display will be coloured black, whereas a partially covered pixel will be coloured as gray whose intensity depends on the line's coverage of the pixel. This technique applied to line shown in figure (c) is shown in figure (d).

We call the technique of setting the intensity proportional to the amount of area covered unweighted area sampling. This technique produces noticeably better results than does setting pixels to full intensity or zero intensity, but there is an even more effective strategy called weighted area sampling.

We must note that unweighted area sampling has the following three properties:- First, the intensity of a pixel intersected by a line edge decreases as the distance between the pixel center and the edge increases. A second property of unweighted area sampling is that the primitive cannot influence the intensity at a pixel at all if the primitive does not intersect the pixel. A third property of the unweighted area sampling is that equal areas contribute equal intensity, regardless of the distance between the pixel's center and the area, only the total amount of overlapped area matters.

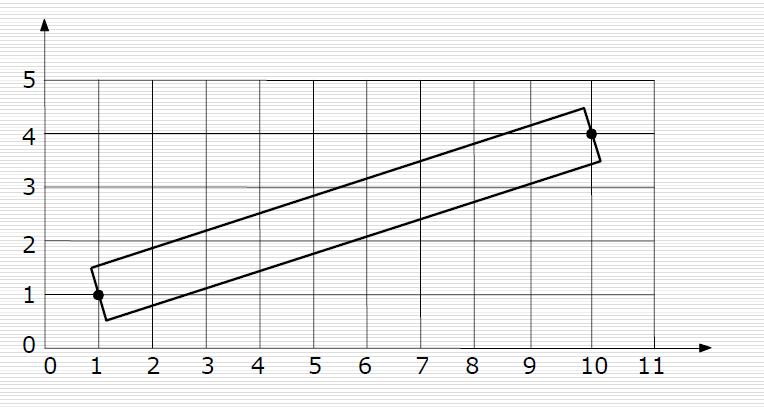


Figure (c)

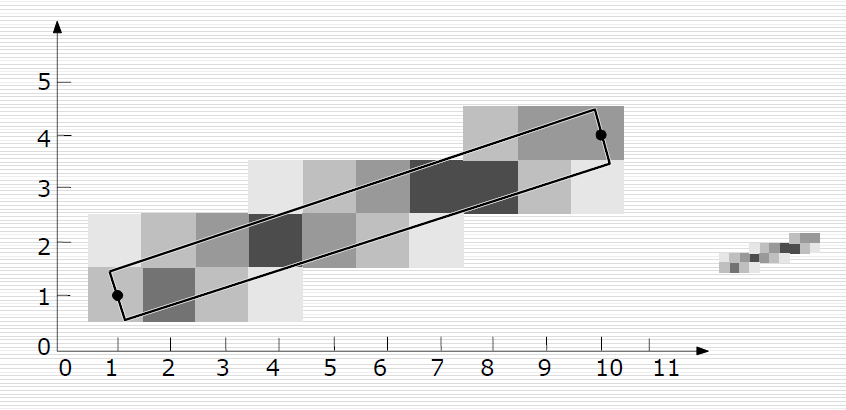
****

Figure (d)

**4. WEIGHTED AREA SAMPLING**

In weighted area sampling, we keep unweighted area sampling's first and second properties but we alter the third property. We let equal areas contribute unequally: A small area closer to the pixel center has a great influence than does one at a greater distance. In our new approach, the pixel represents a circular area larger than the square tile, the primitive will intersect this large, hence, it will contribute to the intensity of the pixel.

To explain the origin of the adjectives unweighted and weighted, we define a weighting function that determines the influence on the intensity of a pixel of a given small area *dA* of a primitive, as a function of *dA*'s distance from the center of the pixel. This function is constant for unweighted area sampling, and decreases with the increasing distance for the weighted area sampling. Think of the weighting function as a function*, W(x,y),* on the plane, whose heighted above the (x,y) plane gives the weight for the area *dA* at *(x,y).* For unweighted area sampling with the pixels represented as square tiles, the graph of *W* is a box.

Square pixels, with centers indicated by crosses at the intersection of grid lines; the weighting function is shown as a box whose base is that of the current pixel. the intensity contributed from all small areas in the region of overlap between the primitive and the pixel. The intensity contributed by each small area is proportional to the area multiplied by the weight. Therefore, the total intensity is the integral of the weighting function over the area of overlap. The volume represented by this integral, *Ws*, is always a fraction between 0 and 1, and the pixel's intensity *I* is *Imax .Ws. Ws* is a wedge of the box. The weighting function is also called a filter function, and the box is also called a box filter. For unweighted area sampling, the height of the box is normalized to 1, so that the box's volume is 1, which causes a thick line covering the entire pixel to have an intensity *I = Imax. I = Imax*. Now let us construct a weighting function for weighted area sampling; it must give less weight to small areas farther away from the pixel center than it does to those closer. We choose a function that has a maximum at the center of the pixel and decreases linearly with increasing distance from the center. Because of rotational symmetry, the graph of this function forms a circular cone. The circular base of the cone should have a radius larger than you might expect. Thus a primitive fairly far from a pixel’s center still has influence on the pixel’s intensity; also, the supports associated with neighbouring pixels overlap, and therefore a single small piece of a primitive may actually contribute to several different pixels. As with the box filter the sum of all the intensity contributions for the cone filter is the volume under the cone and above the intersection of the cone’s base and the primitive; this volume *Ws* is a vertical section of the cone.

## 16. What is open GL? Explain with help of graphics pipeline as well as with a standard program.

OpenGL is a software interface to graphics hardware. This interface consists of about 120 distinct commands, which you use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn't the computer that runs your graphics program. This might be the case if you work in a networked computer environment where many computers are connected to one another by wires capable of carrying digital data.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you must build up your desired model from a small set of geometric primitive - points, lines, and polygons

A Very Simple OpenGL Program. (Have included the whole thing so that you can understand the code, not part of the answer, except the code.)

Because you can do so many things with the OpenGL graphics system, an OpenGL program can be complicated. However, the basic structure of a useful program can be simple: Its tasks are to initialize certain states that control how OpenGL renders and to specify objects to be rendered.

Before you look at an OpenGL program, let's go over a few terms. Rendering, which you've already seen used, is the process by which a computer creates images from models. These *models*, or objects, are constructed from geometric primitives - points, lines, and polygons - that are specified by their vertices.

The final rendered image consists of pixels drawn on the screen; a pixel - short for picture element - is the smallest visible element the display hardware can put on the screen. Information about the pixels (for instance, what color they're supposed to be) is organized in system memory into bitplanes. A bitplane is an area of memory that holds one bit of information for every pixel on the screen; the bit might indicate how red a particular pixel is supposed to be, for example. The bitplanes are themselves organized into a frame buffer, which holds all the information that the graphics display needs to control the intensity of all the pixels on the screen.

### ­­Graphics pipelining

Most modern graphics systems can be thought of as an assembly line, sometimes called a graphics *pipeline*. The main central processing unit (CPU) issues a drawing command, perhaps other hardware does geometric transformations, clipping occurs, then shading or texturing is performed, and finally, the values are written into the bitplanes for display (see Appendix A for details on the order of operations). In high-end architectures, each of these operations is performed by a different piece of hardware that's been designed to perform its particular task quickly. In such an architecture, there's no need for the CPU to wait for each drawing command to complete before issuing the next one. While the CPU is sending a vertex down the pipeline, the transformation hardware is working on transforming the last one sent, the one before that is being clipped, and so on. In such a system, if the CPU waited for each command to complete before issuing the next, there could be a huge performance penalty.

(**ADDITIONAL INFO) :**In addition, the application might be running on more than one machine. For example, suppose that the main program is running elsewhere (on a machine called the client), and that you're viewing the results of the drawing on your workstation or terminal (the server), which is connected by a network to the client. In that case, it might be horribly inefficient to send each command over the network one at a time, since considerable overhead is often associated with each network transmission. Usually, the client gathers a collection of commands into a single network packet before sending it. Unfortunately, the network code on the client typically has no way of knowing that the graphics program is finished drawing a frame or scene. In the worst case, it waits forever for enough additional drawing commands to fill a packet, and you never see the completed drawing.

For this reason, OpenGL provides the command **glFlush()**, which forces the client to send the network packet even though it might not be full. Where there is no network and all commands are truly executed immediately on the server, **glFlush()** might have no effect. However, if you're writing a program that you want to work properly both with and without a network, include a call to **glFlush()** at the end of each frame or scene. Note that **glFlush()** doesn't wait for the drawing to complete - it just forces the drawing to begin execution, thereby guaranteeing that all previous commands execute in finite time even if no further rendering commands are executed.

A few commands - for example, commands that swap buffers in double-buffer mode - automatically flush pending commands onto the network before they can occur. void **glFlush**(void);

Forces previously issued OpenGL commands to begin execution, thus guaranteeing that they complete in finite time.

If **glFlush()** isn't sufficient for you, try **glFinish()**. This command flushes the network as **glFlush()** does and then waits for notification from the graphics hardware or network indicating that the drawing is complete in the framebuffer. You might need to use **glFinish()** if you want to synchronize tasks - for example, to make sure that your three-dimensional rendering is on the screen before you use Display PostScript to draw labels on top of the rendering. Another example would be to ensure that the drawing is complete before it begins to accept user input. After you issue a **glFinish()** command, your graphics process is blocked until it receives notification from the graphics hardware (or client, if you're running over a network) that the drawing is complete. Keep in mind that excessive use of **glFinish()** can reduce the performance of your application, especially if you're running over a network, because it requires round-trip communication. If **glFlush()** is sufficient for your needs, use it instead of **glFinish()**.void **glFinish**(void);

Forces all previously issued OpenGL commands to complete. This command doesn't return until all effects from previous commands are fully realized.