Tulsiramji Gaikwad-Patil College of Engineering and Technology
Wardha Road, Nagpur-441 108
NAAC Accredited

Department of Computer Science & Engineering

Session 2018-19

Semester: V

Subject: Computer Graphics

Summer - 18

Q 1 a. Give various applications of computer graphics. & Explain areas of computer graphics

Computer-Aided Design for engineering and architectural systems etc.

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

Presentation Graphics

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

Computer Art Painting packages are available.

With cordless, pressure-sensitive stylus, artists can produce electronic paintings which simulate different brush strokes, brush widths, and colors. Photorealistic techniques, morphing and animations are very useful in commercial art. For films, 24 frames per second are required. For video monitor, 30 frames per second are required.

Entertainment

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

Education and Training

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

Visualization

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

Image Processing

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

Graphical User Interface

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

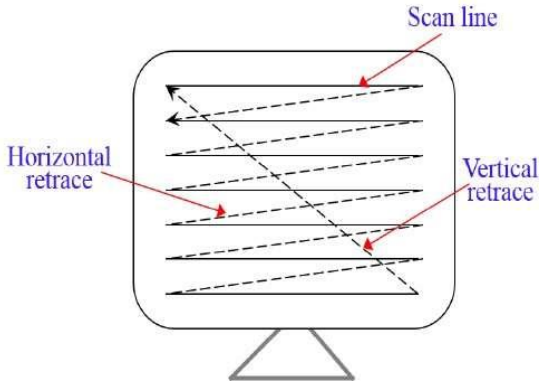
Q 1 b. Differentiate between Raster scan & Random

Raster Scan

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 as shown in the following illustration.

Each screen point is referred to as a pixel (picture element) or pel. 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.

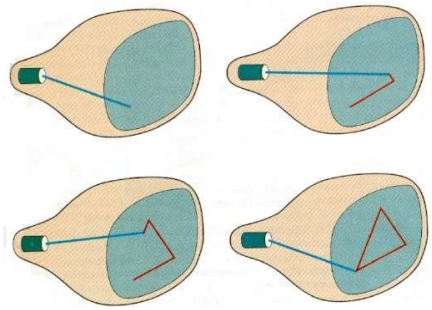


Random Scan (Vector Scan)

In this technique, the electron beam is directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right and top to bottom as in raster scan. It is also called vector display, stroke-writing display, or calligraphic display.

Picture definition is stored as a set of line-drawing commands in an area of memory referred to as the refresh display file. 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 the line-drawing commands are processed, the system cycles back to the first line command in the list.

Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second.



Difference Between Random Scan and Raster 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.
Draw an Image	Screen points/pixels are used to draw an image.	Mathematical functions are used to draw an image.

Q 2 a. Define computer graphics ? Explain working of video controller system?

A **video display controller** or **VDC** (also regularly called **display engine**, **display interface**) is an integrated circuit which is the main component in a video signal generator, a device responsible for the production of a TV video signal in a computing or game system. Some VDCs also generate an audio signal, but that is not their main function.

VDCs were used in the home computers of the 1980s and also in some early video picture systems.

The VDC is the main component of the video signal generator logic, responsible for generating the timing of video signals such as the horizontal and vertical synchronization signals and the blanking interval signal. Sometimes other supporting chips were necessary to build a complete system, such as RAM to hold pixel data, ROM to hold character fonts, or some discrete logic such as shift registers.

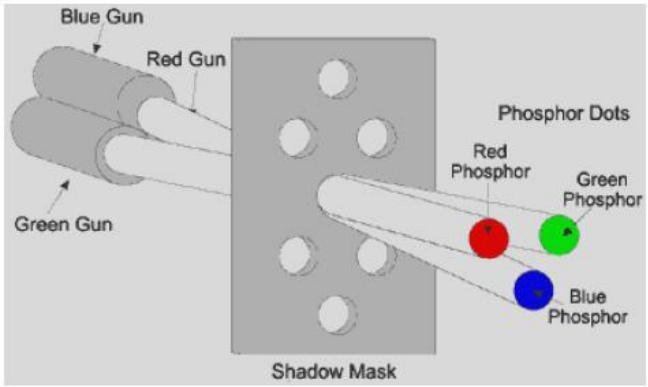
Most often the VDC chip is completely integrated in the logic of the main computer system, (its video RAM appears in the memory map of the main CPU), but sometimes it functions as a coprocessor that can manipulate the video RAM contents independently

Shadow Mask CRT

A common methodology for color CRT display is the Shadow-mask.

In Shadow Mask CRT tiny holes in a metal plate separate the colored phosphors in the layer behind the front glass of the screen. The holes are placed in a manner ensuring that electrons from each of the tube's three cathode guns reach only the appropriately-colored phosphors on the display.

All three beams pass through the same holes in the mask, but the angle of approach is different for each gun. The spacing of the holes, the spacing of the phosphors, and the placement of the guns is arranged so that for example the blue gun only has an unobstructed path to blue phosphors. The red, green, and blue phosphors for each pixel are generally arranged in a triangular shape (sometimes called a "triad").



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 (see doming) images. The invar shadow mask is composed of the nickel-iron alloy invar.

Therefore it expands and contracts much less than other materials in response to temperature changes. This property allows displays made with this technology to provide a clearer, more accurate picture. It also reduces the amount of long-term stress and damage to the shadow mask that can result from repeated expand/contract cycles, thus increasing the display's life expectancy. In other words, In Shadow Mask CRT, before the stream of electrons produced by the CRT's cathode reach the phosphor coated face plate, it encounters the shadow mask, a sheet of metal etched with a pattern of holes.

The mask is positioned in the glass funnel of the CRT during manufacture and the phosphor is coated onto the screen so that electrons coming from the red, green and blue gun positions only land on the appropriate phosphor.

Stray electrons strike the shadow mask and are absorbed by it, generating a great deal of heat, which in turn causes the metal to expand. To allow flatter CRTs to be made, the metal most commonly used now for shadow masks is Invar, an alloy of iron and nickel. The metal has a low coefficient of expansion and its name derives from the supposed invariability of its dimensions when heat is applied.

Q2 b. Write a short note on DVST?

Direct-View Storage Tubes

An alternative method for maintaining a screen image is to store the picture information inside the CRT instead of refreshing the screen. A **direct-view storage tube (DVST)** stores the picture information as a charge distribution just behind the phosphor-coated screen. Two electron guns are used in a DVST. One, the primary gun, is used to store the picture pattern; the second, the flood gun, maintains the picture display.

A DVST monitor has both disadvantages and advantages compared to the refresh CRT. Because no refreshing is needed, very complex pictures can be displayed at very high resolutions without flicker. Disadvantages of DVST systems are that they ordinarily do not display color and that selected parts of a picture cannot be erased. To eliminate a picture section, the entire screen must be erased and the modified picture redrawn. The erasing and redrawing process can take several seconds for a complex picture. For these reasons, storage displays have been largely replaced by raster systems.

Q 3 a. What is DDA Algorithm? What are disadvantages of DDA algorithm? What are advantages of DDA algorithm ?

Advantages of DDA Algorithm

- It is the simplest algorithm and it does not require special skills for implementation.
- It is a faster method for calculating pixel positions than the direct use of equation y=mx + b. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to find the pixel positions along the line path.

Disadvantages

- Floating point arithmetic in DDA algorithm is still time consuming.
- The algorithm is orientation dependent. Hence end point accuracy is poor.
- Although DDA is fast, the accumulation of round-off error in successive additions of floating point increment, however can cause the calculation pixel position to drift away from the true line path for long line segment.

- Rounding-off in DDA is time consuming.

Algorithm :

1. Read the line end points (x1, y1) and (x2, y2) such that they are not equal.
   [if equal then plot that point and exit]

2. Δx = abs(x2 - x1) and Δy = abs(y2 - y1)

3. if($\Delta x \geq \Delta y$) then
   length = Δx
else
   length = Δy
end if

4. Δx = (x2 - x1) / length
   Δy = (y2 - y1) / length

   x = x1 + 0.5
   y = y1 + 0.5

i.= 1

[Begins the loop, in this loop points are plotted]
   while (i $\leq$ length)
   {
    x = x + Δx
    y = y + Δy
    Plot (Integer (x), Integer (y))
    i = i + 1
   }
   End while
7. finish

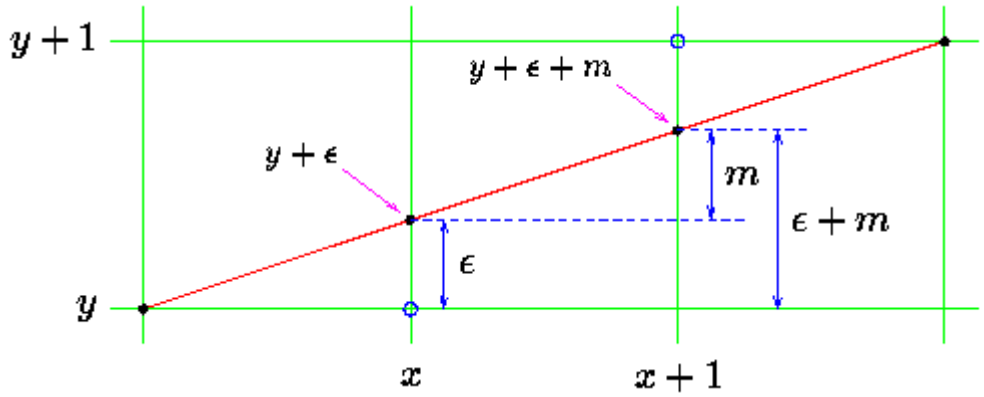
Q 3 b. Draw and Generate Circle in first quadrant

Function for circle-generation

```
// using Bresenham's algorithm
void circleBres(int xc, int yc, int r)
{
   int x = 0, y = r;
   int d = 3 - 2 * r;
   while (y >= x)
   {
      // for each pixel we will
      // draw all eight pixels
      drawCircle(xc, yc, x, y);
      x++;

      // check for decision parameter
      // and correspondingly
      // update d, x, y
      if (d > 0)
      {
         y--;
         d = d + 4 * (x - y) + 10;
      }
      else
         d = d + 4 * x + 6;
      drawCircle(xc, yc, x, y);
      delay(50);
   }
```

c. Bresenhams Line drawing algorithm and advantages



In plotting *(x,y)* the line drawing routine will, in general, be making a compromise between what it would like to draw and what the resolution of the screen actually allows it to draw. Usually the plotted point *(x,y)* will be in error, the actual, mathematical point on the line will not be addressable on the pixel grid. So we associate an error, ϵ, with each *y* ordinate, the real value of *y* should be $y + \epsilon$. This error will range from -0.5 to just under +0.5.

In moving from *x* to *x+1* we increase the value of the true (mathematical) y-ordinate by an amount equal to the slope of the line, *m*. We will choose to plot *(x+1,y)* if the difference between this new value and *y* is less than 0.5.

$$y + \epsilon + m < y + 0.5$$

Otherwise we will plot *(x+1,y+1)*. It should be clear that by so doing we minimise the total error between the mathematical line segment and what actually gets drawn on the display.

The error resulting from this new point can now be written back into ϵ, this will allow us to repeat the whole process for the next point along the line, at *x+2*.

The new value of error can adopt one of two possible values, depending on what new point is plotted. If *(x+1,y)* is chosen, the new value of error is given by:

$$\epsilon_{new} \leftarrow (y + \epsilon + m) - y$$

Otherwise it is:

$$\epsilon_{new} \leftarrow (y + \epsilon + m) - (y + 1)$$

```
ε ← 0,    y ← y1
For x ← x1 to x2 do
    Plot point at (x, y).
    If ( ε + m < 0.5 )
        ε ← ε + m
    Else
        y ← y + 1,    ε ← ε + m - 1
    EndIf
EndFor
```

This still employs floating point values. Consider, however, what happens if we multiply across both sides of the plotting test by Δx and then by 2:

$$\epsilon + m < 0.5$$

$$\epsilon + \Delta y/\Delta x < 0.5$$

$$2\epsilon\Delta x + 2\Delta y < \Delta x$$

All quantities in this inequality are now integral.

Q 5 b. Explain Level of Abstraction In OPEN GL

OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geomteric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primatives, together with commands that control how these objects are rendered (drawn).

Since OpenGL drawing commands are limited to those that generate simple geometric primitives (points, lines, and polygons), the OpenGL Utility Toolkit (GLUT) has been created to aid in the development of more complicated three-dimensional objects such as a sphere, a torus, and even a teapot. GLUT may not be satisfactory for full-featured OpenGL applications, but it is a useful starting point for learning OpenGL.

GLUT is designed to fill the need for a window system independent programming interface for OpenGL programs. The interface is designed to be simple yet still meet the needs of useful OpenGL programs. Removing window system operations from OpenGL is a sound decision because it allows the OpenGL graphics system to be retargeted to various systems including powerful but expensive graphics workstations as well as mass-production graphics systems like video games, set-top boxes for interactive television, and PCs.

GLUT simplifies the implementation of programs using OpenGL rendering. The GLUT application programming interface (API) requires very few routines to display a graphics scene rendered using OpenGL. The GLUT routines also take relatively few parameters.

c. Explain feature of OPENGL

1. **Alpha Blending**

Provides a means to create transparent objects. Using alpha information and object can be defined as anything from totally transparent to totally opaque.

2. **Color – index mode**

Color buffers store color indices rather than red, green, blue and alpha color components.

3. **Display list**

The contents of a display list may be pre-processed and might therefore execute more efficiently than the same set of OpenGL commands executed in immediate mode.

4. **Double buffering**

Used to provide smooth animation of objects. Each successive scene of an object in motion can be constructed in the back or 'hidden' buffer and the n displayed. This allows only complete images to ever be displayed on the screen.

5. **Feedback**

A mode where OpenGL will return the processed geometric information (colors, pixel positions etc.) To the application as comported to rendering them into the frame buffer.

6. **Immediate mode**

Execution of OpenGL commands when they are called rather than from a display list.

7. **Materials lighting and shading**

The ability to accurately compute the color of any point given the material properties for the surface.

8. **Pixel operations**
Storing, transforming, mapping and zooming.

9. **Polynomial evaluators**
To support non-uniform rational B-splines (NURBS).

10. **Selection and Picking**
A mode in which OpenGL determine whether certain user-identified graphics primitives are rendered into a region of interest in the frame buffer.

11. **Texture mapping**
The process of applying an image to graphics primitive. The technique is used to generate realism in images.

12. **Z- buffering**
The Z- buffering is used to keep track of whether one part of an object is closer to the viewer than another it is important in hidden surface removal.

Any visual computing application requiring maximum performance-from 3D animation to CAD to visual simulation – can exploit high-quality, high-performance OpenGL capabilities. These capabilities allow developers in diverse markets such as broadcasting, CAD/CAM/CAE, entertainment, medical imaging, and virtual reality to produce and display incredibly compelling 2D and 3D graphics.

Q 6 b. Draw and Explain operation in OPEN GL

OpenGL Operations

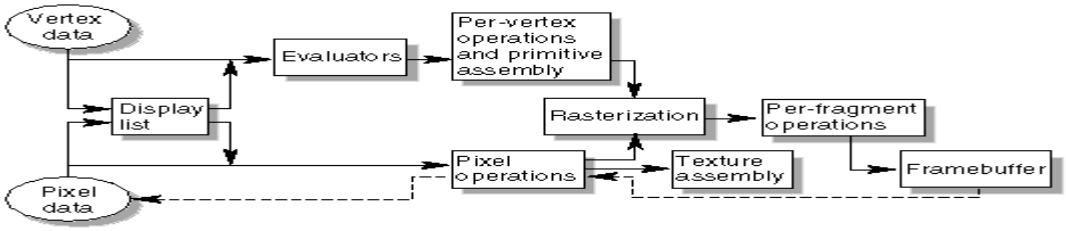


Fig. Order of Operations

Display Lists

All data, whether it describes geometry or pixels, can be saved in a display list for current or later use. (The alternative to retaining data in a display list is processing the data immediately - also known as immediate mode.) When a display list is executed, the retained data is sent from the display list just as if it were sent by the application in immediate mode.

Evaluators

All geometric primitives are eventually described by vertices. Parametric curves and surfaces may be initially described by control points and polynomial functions called basis functions. Evaluators provide a method to derive the vertices used to represent the surface from the control points. The method is a polynomial mapping, which can produce surface normal, texture coordinates, colors, and spatial coordinate values from the control points.

Per-Vertex Operations

For vertex data, next is the "per-vertex operations" stage, which converts the vertices into primitives. Some vertex data (for example, spatial coordinates) are transformed by 4 x 4 floating-point matrices. Spatial coordinates are projected from a position in the 3D world to a position on your screen.

Primitive Assembly

Clipping, a major part of primitive assembly, is the elimination of portions of geometry which fall outside a half-space, defined by a plane. Point clipping simply passes or rejects vertices; line or polygon clipping can add additional vertices depending upon how the line or polygon is clipped.

In some cases, this is followed by perspective division, which makes distant geometric objects appear smaller than closer objects. Then viewport and depth (z coordinate) operations are applied. If culling is enabled and the primitive is a polygon, it then may be rejected by a culling test. Depending upon the polygon mode, a polygon may be drawn as points or lines.

Pixel Operations

While geometric data takes one path through the OpenGL rendering pipeline, pixel data takes a different route. Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components. Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the rasterization step.

If pixel data is read from the frame buffer, pixel-transfer operations (scale, bias, mapping, and clamping) are performed. Then these results are packed into an appropriate format and returned to an array in system memory.

There are special pixel copy operations to copy data in the frame buffer to other parts of the frame buffer or to the texture memory. A single pass is made through the pixel transfer operations before the data is written to the texture memory or back to the frame buffer.

Texture Assembly

An OpenGL application may wish to apply texture images onto geometric objects to make them look more realistic. If several texture images are used, it's wise to put them into texture objects so that you can easily switch among them.

Some OpenGL implementations may have special resources to accelerate texture performance. There may be specialized, high-performance texture memory. If this memory is available, the texture objects may be prioritized to control the use of this limited and valuable resource.

Rasterization

Rasterization is the conversion of both geometric and pixel data into *fragments*. Each fragment square corresponds to a pixel in the frame buffer. Line and polygon stipples, line width, point size, shading model, and coverage calculations to support antialiasing are taken into consideration as vertices are connected into lines or the interior pixels are calculated for a filled polygon. Color and depth values are assigned for each fragment square.

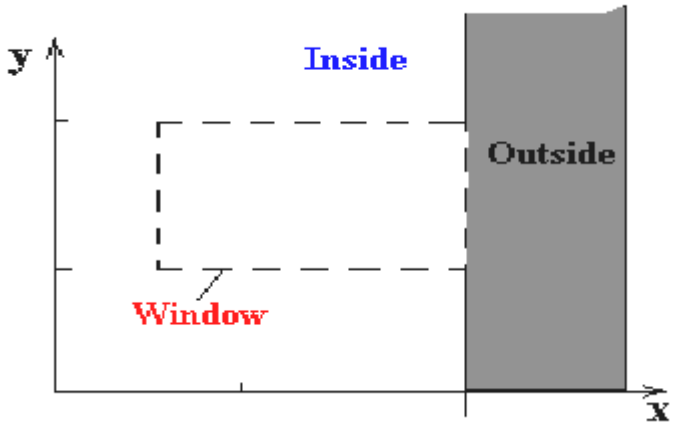
Fragment Operations

Before values are actually stored into the frame buffer, a series of operations are performed that may alter or even throw out fragments. All these operations can be enabled or disabled.

The first operation which may be encountered is texturing, where a texel (texture element) is generated from texture memory for each fragment and applied to the fragment. Then fog calculations may be applied, followed by the scissor test, the alpha test, the stencil test, and the depth-buffer test (the depth buffer is for hidden-surface removal). Failing an enabled test may end the continued processing of a fragment's square. Then, blending, dithering, logical operation, and masking by a bitmask may be performed.

Q 8 a. Explain Sutherland Cohen algorithm ?

To determine whether endpoints are inside or outside a window, the algorithm sets up a **half-space code** for each endpoint. Each edge of the window defines an infinite line that divides the whole space into two half-spaces, the **inside half-space** and the **outside half-space**, as shown below.



As you proceed around the window, extending each edge and defining an inside half-space and an outside half-space, nine regions are created - the eight "outside" regions and the one "inside"region. Each of the nine regions associated with the window is assigned a 4-bit code to identify the region. Each bit in the code is set to either a **1**(true) or a **0**(false). If the region is to the **left** of the window, the **first** bit of the code is set to 1. If the region is to the **top** of the window, the **second** bit of the code is set to 1. If to the **right**, the **third** bit is set, and if to the **bottom**, the **fourth** bit is set. The 4 bits in the code then identify each of the nine regions as shown below.

1001	0001	0101
1000	0000 Window	0100
1010	0010	0110

For any endpoint **(x , y)** of a line, the code can be determined that identifies which region the endpoint lies. The code's bits are set according to the following conditions:

- First bit set **1** : Point lies to **left** of window $x < x_{min}$
- Second bit set **1** : Point lies to **right** of window $x > x_{max}$
- Third bit set **1** : Point lies below(**bottom**) window $y < y_{min}$
- fourth bit set **1** : Point lies above(**top**) window $y > y_{max}$

The sequence for reading the codes' bits is **LRBT** (Left, Right, Bottom, Top).

Once the codes for each endpoint of a line are determined, the logical **AND** operation of the codes determines if the line is completely outside of the window. If the logical AND of the endpoint codes is **not zero**, the line can be trivally rejected. For example, if an endpoint had a code of 1001 while the other endpoint had a code of 1010, the logical AND would be 1000 which indicates the line segment lies outside of the window. On the other hand, if the endpoints had codes of 1001 and 0110, the logical AND would be 0000, and the line could not be trivally rejected.

The logical **OR** of the endpoint codes determines if the line is completely inside the window. If the logical OR is **zero**, the line can be trivally accepted. For example, if the endpoint codes are 0000 and 0000, the logical OR is 0000 - the line can be trivally accepted. If the endpoint codes are 0000 and 0110, the logical OR is 0110 and the line can not be trivally accepted.

Q 8 b. Explain midpoint subdivision algorithm in detail.

Read two end points of line P1 (x1,y1) and P2 (x2,y2).
2) Read corners of window (Wx1, Wy1) and (Wx2, Wy2).
3) Assign region codes for P1 and P2. A region code is a 4 digit bit code which indicates one of nine regions having the end point of line.
Initialize code with bits 0000
Set Bit 1 if (x<Wx1) Set Bit 2 if (x>Wx2)
Set Bit 3 if (y<Wy2) Set Bit 4 if (y>Wy1)
4) Check visibility of line P1P2
a) If region codes for both end points P1 and P2 are 0 = > Line is completely visible. Draw Line and Go to Step 3.
b) If region codes for both end points P1 and P2 are not 0 and logical ANDing of them is also not zero = > Line is completely invisible. Reject line and Go to Step 3.
c) If a) and b) both are not satisfied, line is partially visible.
5) Divide partially visible line segments to equal parts and repeat steps 3 through 5 for both sub divided line segments until you get completely visible and completely invisible line segments.
6) Stop.